Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería en Electrónica y Automática Industrial

Trabajo Fin de Grado

Controlador MPC adaptativo por DQN orientado a la conducción autónoma

Autor: Jesús Ángel Oliveros Fernández

Tutor: Iván García Daza

2021

UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería en Electrónica y Automática Industrial

Trabajo Fin de Grado

Controlador MPC adaptativo por DQN orientado a la conducción autónoma

Autor: Jesús Ángel Oliveros Fernández

Director: Iván García Daza

Tribunal:

Presidente: Javier de Pedro Carracedo

Vocal 1º: Javier Alonso Ruiz

Vocal 2º: Iván García Daza

Calificación: .....

Fecha: .....

Este proyecto se lo dedico a la persona esencial de mi vida...

"Dedicado a ti, mamá, gracias por todo..."

Jesús Ángel Oliveros Fernández.

Agradecimientos

Dedicado a ti, mamá, gracias por todo...

En primer lugar quería dar las gracias a mi tutor de este TFG, Iván García Daza, por depositar toda su confianza en mi para realizar este proyecto. Iván, te agradezco mucho que hayamos podido llevar a cabo este gran trabajo juntos de la mano en cada una de las fases del mismo. ¿Recuerdas por los duros momentos que pasamos al principio hasta que conseguimos arrancar? Quiero darte las gracias por hacer todo esto mucho más llevadero con la relación tutor - alumno tan buena y cercana que se creó entre nosotros, sabiendo ayudarnos en las situaciones más complicadas y charlando en varias reuniones sobre las rutas por la Alcarria que hacíamos cada uno los fines de semana. Además, también quería agradecerte la oportunidad que me brindaste del contrato de investigación durante los primeros meses de este proyecto, fue una experiencia muy buena que pude vivir y disfrutar con los compañeros del laboratorio de investigación INVETT y mi compañera Lucía.

Tengo que dar las gracias a la universidad por darme el acceso a conocer a personas maravillosas. Pero en especial, quiero mencionar y agradecer a mi grupo "Los Reales". Partiendo de una escapada en invierno a Cercedilla en primero de carrera, hemos hecho planes que jamás pensé que haría con gente que conocía desde tan poco tiempo. Con solo 6 meses de carrera, nos fuimos a Altea, después de casa rural, al año siguiente de viaje a Ámsterdam, más casas rurales...etc. Además de aquellas noches en las que celebrábamos, principalmente por Alcalá, el fin de alguna etapa de exámenes como si fuese la última noche del año. Todos estos momentos y cada una de las personas que formáis este grupo, para mi habéis sido esenciales para llevar a cabo de la mejor forma posible esta carrera y llegar hoy aquí a redactar esto. Tengo muy claro que sin vosotros, esto no habría sido lo mismo.

Quiero agradecer también a mis amigos de Marchamalo por apoyarme y aguantar mis agobios durante todos estos años. En especial, a aquellos que para mi han sido más que unos amigos, unos hermanos, me habéis abierto las puertas de vuestras casas y familias en los momentos más duros para mi, de nuevo, una vez más, gracias.

A mi familia, agradecer toda la confianza que han depositado en mi en estos años y durante toda mi formación. En especial, quiero agradecer a mi Primo David y su novia Laura, todo lo que han hecho por mi, porque sabéis de sobra, que ambos sois los responsables de que ahora mismo esté escribiendo esto para graduarme de una ingeniería. También quiero dar las gracias a mi Prima Estefanía por enseñarme todas sus habilidades relacionadas con el inglés y las entrevistas de trabajo y recibirme en Mánchester cuando estuvo viviendo allí para aprender inglés, gracias cousin. A mi prima Laura Oliveros quiero darle las gracias por enseñarme a cogerle cariño a la química durante el primer año de carrera, ya sabes prima, siempre serás mi mejor profesora. Quiero daros las gracias también a ti Papá y a Ana, por vuestro apoyo y enseñarme lecciones de vida que me han hecho aprender.

Quería dejar para lo último lo que ha sido lo más importante. Las personas que han convivido conmigo y me han aguantado estos cuatro años, día a día, en los mejores y en los peores momentos. Mamá, Juan Carlos y Julián. Vosotros me habéis visto reir, llorar, sufrir, disfrutar, pero sobre todo, aprender, día a día a ser una persona más fuerte para llevar con fuerza esta carrera. No se cómo os puedo agradecer todo el apoyo que me habéis dado y la confianza que habéis depositado en mi para lograr alcanzar este objetivo de graduarme. Si hay alguien que sabe realmente el esfuerzo que me ha costado esta carrera, sois vosotros. Una vez más gracias por todo. Juan Carlos, mi hermano, aunque seamos el día y la noche, quiero que sepas que todo lo que me has enseñado ha sido esencial para aprender a ser más fuerte día a día. Julián, gracias por toda la confianza y esfuerzo que has depositado en mi. Mamá, una vez más, al igual que comencé los agradecimientos los termino, este proyecto va dedicado a ti, por ser la persona más importante de mi vida y cumplir juntos este sueño, gracias.

Jesús Ángel Oliveros Fernández

Resumen

El empleo de técnicas de aprendizaje por refuerzo para optimizar sistemas de conducción autónoma está en constante crecimiento en la actualidad. Los objetivos de este proyecto están basados en mejorar el comportamiento de un controlador MPC de trayectorias de un vehículo autónomo mediante el uso de técnicas de aprendizaje por refuerzo para estimar el valor óptimo de los pesos de la función de costes del MPC asociados al error de distancia lateral y de yaw.

El proyecto se inicia desde una versión del controlador MPC en la que los pesos de error de distancia lateral y de yaw (wd y wy respectivamente) tomaban ambos un valor de 10000. Estos valores obtenidos experimentalmente no presentaban errores altos, aunque aun así se propuso encontrar los valores de pesos óptimos que consiguiesen mejorar el comportamiento del vehículo reduciendo en mayor parte ambos errores a lo largo de la trayectoria.

Tras un estudio teórico de las técnicas de aprendizaje por refuerzo, se escogió el algoritmo Actor - Critic por sus similitudes de funcionamiento con las características del problema a resolver. Inicialmente se realizaron una serie de ensayos sobre el entorno de simulación CartPole para entender su funcionamiento y posteriormente implementarlo sobre el sistema del controlador MPC y analizar su comportamiento en el entorno de simulación de vehículos CARLA.

El peso asociado al error de distancia lateral fue el primero en ser optimizado buscando así unos valores de este peso que consiguieran reducir lo máximo posible dicho error. Después, fue modificada la arquitectura del MPC y hubo que adaptar el algoritmo Actor - Critic a esta nueva versión para estimar unos nuevos valores de pesos óptimos de error de distancia lateral.

Por último, de forma similar, en la versión final del controlador MPC se procedió a optimizar el error de yaw del vehículo de nuevo con el algoritmo Actor - Critic obteniendo un rango de valores de peso de error de yaw que consiguiesen reducir su error respecto a la versión original del MPC.

En ambos casos se obtienen unos resultados bastante favorables que consiguen reducir significativamente los errores originales de distancia lateral y de yaw del MPC respecto a su versión original, además de ofrecer una respuesta mucho más suave del sistema, resultado así de una conducción más agradable y segura para los integrantes del vehículo.

Palabras clave: controlador, optimización, error, lateral, yaw.

Abstract

The use of Reinforcement Learning techniques to optimize autonomous driving systems is constantly increasing nowadays. The objectives of this project are based on improving the performance of an MPC controller of trajectories of an autonomous vehicle by using Reinforcement Learning techniques to estimate the optimal value of the weights of the cost function of the MPC related to the lateral distance and yaw error.

The project begins since an initial MPC controller version in which the weights of lateral distance and yaw (wd and wy respectively) were both 10000. These values experimentaly obtained do not reach to high errors, although it was proposed to search the optimal values of them that will get a better performance of the vehicle decreasing as much as posible both errors over all the trajectory.

After a theoretical study of the different Reinforcement Learning techniques, it was chosen the Actor - Critic algorithim due to the similarities of its operation characteristics according to the problem to solve. At the beginning, there were tested some trials over the CartPole enviroment in order to understand its performance and then code it in the complete system with the MPC controller and analyze its performance over the simulation enviroment of autonomous vehicles CARLA.

The weight related to the lateral distance error was the first to be optimized by searching the weights values that decreases it as much as possible. Afterwards, it was modified the architecture of the MPC controller and the Actor - Critic algorithim had to be modified also in order to adapt it to the new version of the MPC to search the new optimal weight values of lateral distance error.

Finally, in a similar way, in the last version of the MPC controller there was obtained also with the Actor - Critic algorithim the optimal values of the weight associated to the yaw error that decrease as much as possible this error respect to the original version of the MPC.

In both cases there were obtained excellent results that significantly decrease the original errors of lateral distance and yaw of the MPC, besides their offer an smoothness system response that leads to a more confortable and secure driving for the integrals of the vehicle.

Keywords: controller, optimization, error, lateral, yaw.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1 Marco del proyecto

La conducción autónoma ya no es el futuro, es el presente. Con la constante mejora de los sistemas de transporte se espera una reducción considerable en los accidentes de tráfico hasta alcanzar cifras prácticamente nulas en las próximas décadas. Los vehículos autónomos poseerán mayor seguridad y confortabilidad en la conducción en comparación con la experiencia actual de conducción humana, pues los sensores, unidades de procesamiento y control y conectividad entre toda la red de vehículos serán capaces de suministrar la información necesaria para llegar a un nivel de conducción lo más segura y eficiente posible.

Numerosas universidades y empresas del sector automovilístico están invirtiendo en grandes proyectos de investigación para el desarrollo de esta tecnología. En un futuro no muy lejano se espera un nivel de conducción autónoma 5, el cual es considerado el mayor de todos, en que el vehículo será completamente autónomo y no se requerirá en absoluto de ningún tipo de acción humana. Aún quedan unos años para ver este fenómeno, aunque diversas marcas líderes del sector como Tesla, invierten la mayor parte de su actividad empresarial en esta tecnología. Actualmente, esta compañía posee a la venta vehículos autónomos con nivel de conducción autónoma 2, en los que el ser humano tiene que estar totalmente presente en la conducción, pero el vehículo es capaz de recoger información del entorno y circular por sí mismo, así como de realizar cambios de carril y adelantamientos únicamente indicándole con el intermitente el sentido del desplazamiento por parte del usuario.

La seguridad en la conducción de vehículos está compuesta por diversos factores como el estado tráfico, las condiciones geométricas y climatológicas del entorno y el factor humano, considerado el principal causante de la mayoría de los accidentes de tráfico. En el área de investigación de la conducción autónoma hay dos grandes universos por explorar, entre los que se incluyen la investigación acerca de la optimización de los factores ajenos al vehículo que intervienen en la conducción como podrían ser la mejora en el diseño de carreteras y por otro lado el propio estudio del vehículo. [1] Este proyecto se centrará en la investigación del vehículo autónomo como tal, dentro de su vertiente de control.

En el control de la trayectoria de un vehículo autónomo, serán necesarios los siguientes elementos: percepción del entorno, conocimiento previo de la trayectoria deseada y por último control del sistema y actuadores del vehículo.

Este proyecto se centrará en la parte de control del sistema. La percepción del vehículo y trayectoria deseada así como los actuadores del vehículo se asumirá que se reciben y envían los datos procesados por

el controlador y se focalizará el desarrollo de la parte del control del sistema para seguir la trayectoria deseada por el vehículo.

Como en todo sistema de control, para llevar a cabo la tarea de control es necesario el conocimiento de la planta del sistema. El conjunto de elementos de un vehículo así como su geometría hacen que su modelado adquiera bastante complejidad. Es por esta razón por la que se opta por desarrollar un controlador neuronal predictivo basado en modelo, comúnmente conocido como MPC.

El controlador MPC diseñado y validado en simulación en proyectos anteriores se toma como punto de partida de este proyecto. Actualmente, este controlador posee un funcionamiento aceptable sobre el entorno de simulación CARLA y está siendo testeado en el vehículo real automatizado *Citroën C4* del grupo de investigación INVETT *(INtelligent VEhicules and Traffic Technologies) [2]* del departamento de Automática de la Universidad de Alcalá, dentro del cual este proyecto es desarrollado.

Aunque el comportamiento del vehículo por la trayectoria de prueba, controlado por el MPC, es aceptable, **el objetivo fundamental de este proyecto es optimizar su funcionamiento con algoritmos de aprendizaje por refuerzo.** Dicha optimización se basará en reducir los errores de distancia lateral y de orientación (yaw) del vehículo. El valor de estos errores dependen fundamentalmente de los pesos asociados a cada uno de los errores en la función de costes del controlador MPC, por lo que el proyecto se centrará en **encontrar los valores de pesos de distancia lateral y de yaw óptimos del MPC para minimizar sus correspondientes errores gracias al empleo de técnicas de aprendizaje por refuerzo**. El desarrollo del algoritmo será desarrollado en Python y validado en el simulador CARLA para en un futuro poder implementarlo en el vehículo real del grupo de investigación.

1.2 Controlador predictivo basado en modelo (MPC)

Un MPC *(Model Predictive Control)* de las siglas en inglés, es un controlador neuronal multivariable que genera señales de control futuras dentro de un horizonte temporal finito. [3] Su objetivo es controlar un sistema mediante la optimización de una función de costes o error dentro de unas restricciones impuestas. En la siguiente imagen se muestra una gráfica de la predicción de acciones futuras generadas por un MPC:

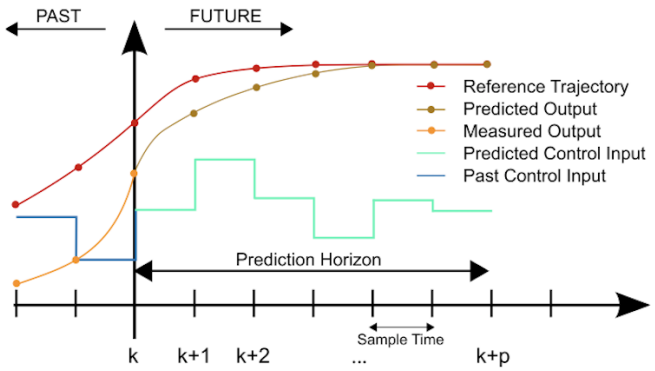


Figura 1.1: Funcionamiento de un MPC

Como se puede observar en la figura, el MPC predice la salida y la señal de control en función de la referencia de entrada para un horizonte temporal de predicción finito.

[4] El diseño de un controlador MPC para el seguimiento de trayectorias en un vehículo autónomo se justifica en que en la conducción se deben tener calculadas unas acciones de control futuras en un horizonte temporal finito, pues no sería una conducción segura aquella en la que para cada instante temporal se fuera generando una señal de control instantánea de acuerdo a la entrada de la planta del sistema, ya que, por ejemplo, en el caso de una trayectoria tipo curva, se debe actuar con anterioridad sobre la dinámica del vehículo para tomar correctamente la curva, pues es físicamente imposible reducir la velocidad y orientar la dirección de un vehículo de forma inmediata. Es así como el controlador MPC trata de comportarse de acuerdo a una conducción real con la ventaja ser capaz de seguir una trayectoria de referencia de la forma más óptima posible.

En este proyecto se parte de un controlador MPC previamente diseñado para el seguimiento de la trayectoria de un vehículo autónomo. El funcionamiento de este algoritmo se basa en la optimización de una función de costes, reduciendo lo máximo posible el error entre la referencia de la trayectoria que el vehículo debe seguir y la salida que en controlador genera de acuerdo al modelo dinámico del vehículo. La función de costes de un MPC *(J)* está compuesta principalmente por la suma de los errores cinemáticos y dinámicos de la planta del sistema elevados al cuadrado, multiplicados cada uno de ellos por unos coeficientes denominados **pesos (w)** para cada instante (i) del horizonte temporal (N).

La función de costes *(J)* del controlador MPC empleado en este proyecto, está compuesta por los errores del estado del controlador, formado por el error de distancia lateral ***ed***, error del ángulo yaw del vehículo ***ey*** y error de velocidad ***ev***. Además, para garantizar una conducción confortable, se incluyen dos términos más correspondientes con la variación de aceleración y del *steering* o giro del volante, denotados con Δ_Θ y $\Delta\delta_f$ respectivamente. En la ecuación 1.1 se muestra la función de costes *(J)* del controlador MPC.

$$J = \sum_{i=1}^{N} e_d(k+i)^T w_d e_d(k+i) + \sum_{i=1}^{N} e_y(k+i)^T w_y e_y(k+i) + \sum_{i=1}^{N} e_v(k+i)^T w_v e_v(k+i) + \sum_{i=0}^{N-1} \Delta_\Theta(k+i)^T w_{\Delta_\Theta} \Delta_\Theta(k+i) + \sum_{i=0}^{N-1} e_{\Delta\delta_f}(k+i)^T w_{\Delta\delta_f} e_{\Delta\delta_f}(k+i) \tag{1.1}$$

Cada término de error está multiplicado por un peso (w) que atribuye mayor o menor importancia al error asociado. **El objetivo fundamental de este proyecto será encontrar los valores óptimos de los pesos asociados al error de distancia lateral (wd) y al error de yaw (wy) del seguimiento de la trayectoria del vehículo autónomo.**

Como una primera aproximación en el diseño del MPC se establecieron unos valores experimentales a estos pesos. Para obtener los valores óptimos de estos pesos y así reducir los errores asociados a los mismos, se empleará un algoritmo neuronal de **aprendizaje por refuerzo,** en concreto el algoritmo **Actor - Critic.**

1.3 Aprendizaje por refuerzo (Reinforcement Learning)

[5] [6] El aprendizaje por refuerzo o Reinforcement Learning en inglés (RL en adelante), es una técnica de aprendizaje neuronal en la que sobre los estados (s) de un entorno, un agente toma acciones (a) que provocan una transición a un nuevo estado (s') recibiendo una recompensa (r) que evalúa cómo de buena fue la acción tomada. En la siguiente imagen 1.2 [6] se muestra un ejemplo gráfico de los fundamentos del RL:

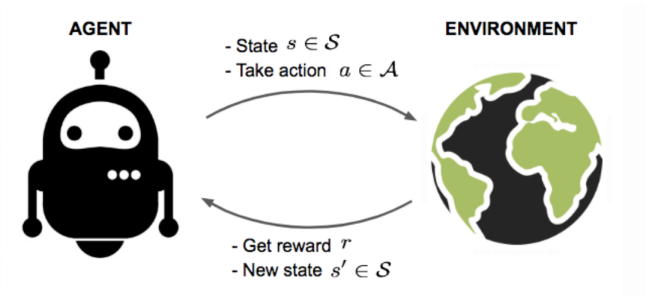


Figura 1.2: Ejemplo gráfico de los fundamentos del RL

[7] [8] Son muchos los campos en los que se están empleando algoritmos de RL al demostrar verdadera efectividad en la resolución de diversos problemas entre los que se describen algunos campos de aplicación a continuación:

- **Sistemas de navegación.** Aplicaciones en sistemas de posicionamiento y seguimiento de trayectorias en robots, drones y vehículos autónomos.
- **Gestión de recursos en procesos industriales como sistemas de climatización y gestión de stock.**
- **Tratamientos de salud.** En los que se han optimizado el tipo de medicamentos y dosis a administrar.
- **Operaciones de márketing.** Estudio sobre campañas de márketing online y digital acerca del modelo que mejor se adapta al cliente.
- **Brazos robóticos.** El caso del sistema *Dactyl* desarrollado por Open AI que consiste en posicionar un cubo en la orientación deseada por el usuario mediante una mano robótica sin tener ningún conocimiento previo del entorno. Se muestra el ejemplo en la imagen a continuación:

Figura 1.3: Ejemplo de aplicación de RL en sistema Dactyl

- **Videojuegos.** Son muchos los videojuegos que han sido empleados durante el desarrollo de sistemas de inteligencia artificial para poner a prueba el aprendizaje neuronal. Un caso específico corresponde con el juego *AlphaStar* que previamente fue entrenado con aprendizaje supervisado, para posteriormente mejorar sus estrategias de juego mediante técnicas de aprendizaje por refuerzo. Se muestra una imagen del entorno del juego *AlphaStar* a continuación:

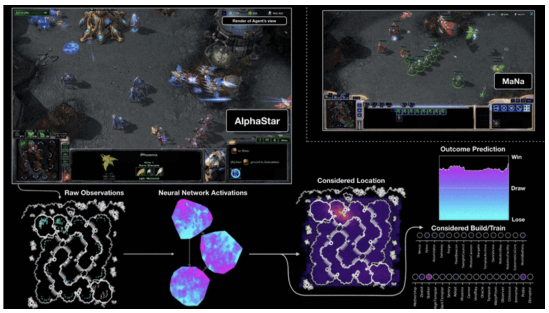


Figura 1.4: Ejemplo de aplicación de RL en el juego AlphaStar

Dentro de las técnicas de entrenamiento de redes neuronales artificiales, esta técnica de aprendizaje se sitúa en el aprendizaje no supervisado, pues no se dispone de una base de datos con las salidas esperadas o deseadas para la red, si no que solo se muestran entradas a la red y el sistema debe ser capaz de estimar las salidas futuras en función del estado actual (entrada de la red) y una política de rewards.

[6] Este concepto puede asimilarse a un proceso de decisión Markov, en inglés Markov Decision Process (MDP), que consistente en que, para cada muestra de entrenamiento, en función del estado anterior, estado actual y la acción tomada por la red, se obtiene una recompensa que evalúa cómo de buena fue la acción tomada. La característica principal de un proceso Markov consiste en que el futuro únicamente depende del estado actual. Es decir, las acciones pasadas y futuras de un proceso son independientes en el sentido de que solo se emplea el estado actual para establecer la probabilidad de las acciones futuras a escoger por el agente sobre el entorno. El diagrama de bloques que muestra el funcionamiento del RL se muestra en la Figura 1.5 [6] a continuación:

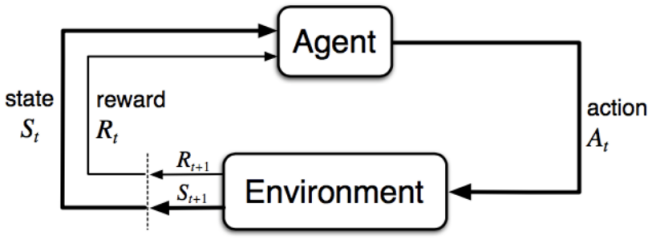


Figura 1.5: Diagrama de bloques del funcionamiento del RL

[6] El objetivo del aprendizaje por refuerzo es obtener el número máximo de rewards en un episodio de entrenamiento a través del aprendizaje obtenido de los episodios pasados, mejorando así las acciones que el agente deberá tomar sobre el entorno para maximizar el número de rewards por episodio.

[6] El funcionamiento del RL consiste en que el agente interacciona con el entorno provocando una transición de estados. La forma en la que el entorno reacciona a las acciones tomadas por el agente, está definida por el modelo del entorno que puede ser conocido o desconocido. La acción que el agente toma para transicionar de estado viene dada por una probabilidad (P) asociada a cada una de las acciones para

obtener así una buena recompensa (reward) por la acción tomada de acuerdo a la ecuación que muestra a continuación:

$$P(s', r|s, a) = \mathbb{P}[S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a] \tag{1.2}$$

[6] El agente posee una política π que evalúa su comportamiento, y sirve para escoger la acción a tomar en el estado actual pudiendo ser determinística o estocástica. En este proyecto se empleará una política estocástica para el agente, asociando así una probabilidad a cada una de las posibles acciones para el estado actual.

[6] Cada estado del sistema está asociado con una función denominada *valor* que se encarga de predecir las recompensas (rewards) futuras que serán recibidas por las acciones tomadas en el estado actual. Esta función de valor se aproxima a una suma de los rewards por muestra en un episodio de entrenamiento descontando los futuros por un factor γ de acuerdo con la ecuación que se muestra a continuación:

$$G_t = R_{t+1} + \gamma R_{t+2} + ... \tag{1.3}$$

El factor de descuento γ perteneciente al intervalo [0,1] tiene como objetivo penalizar los rewards futuros ya que se tiene más conocimiento de los rewards actuales que de los futuros. Cuanto más cercano esté a 1 este valor, mayor será su contribución al descuento de rewards futuros en la función valor del algoritmo de RL.

Tanto la política del agente como la función valor son los datos que se emplearán para generar ese *target* del aprendizaje supervisado sobre el que realizar el aprendizaje de la red neuronal para optimizar el proceso.

Se define como la función acción-valor aquella que se encarga de asociar un valor Q a cada par estado-acción, en similitud con el clásico Q-learning [9] de acuerdo a la ecuación que se muestra a continuación [6]:

$$Q_\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a] \tag{1.4}$$

Combinando el Q-learning con el empleo de apredizaje profundo mediante el uso de redes neuronales *fully-connected*, es de donde se obtiene el algoritmo DQN sobre el que se propuso emplear su funcionamiento para optimizar el controlador MPC en este proyecto. [10]. El algoritmo DQN parte de los fundamentos del Q - learning, en el que se asigna un valor Q a cada par estado - acción y para cada transición de estados se asigna un valor para cada acción a través de una arquitectura neuronal para después escoger el máximo de todos los posibles que será aquel que provoque la mejor acción del agente sobre el entorno. Este algoritmo es válido para un sistema en el que con un conjunto de acciones finito se obtengan unos resultados apropiados. Las limitaciones de este algoritmo se muestran cuando se plantea un entorno complejo sobre el que es necesario disponer de un número muy elevado de acciones a tomar para resolver el problema planteado. Para resolver esta limitación del algoritmo DQN se plantea un nuevo método denominado Actor - Crítico *(en adelante Actor - Critic en ingles)* que consiste en separar por un lado la parte que se encarga de escoger las acciones del agente sobre el entorno y por otro lado de producir los valores Q para cada par estado - acción, obteniendo así un número ilimitado de valores Q. En la Figura 1.6 [10] se muestra un esquema gráfico del funcionamiento del algoritmo Actor - Critic:

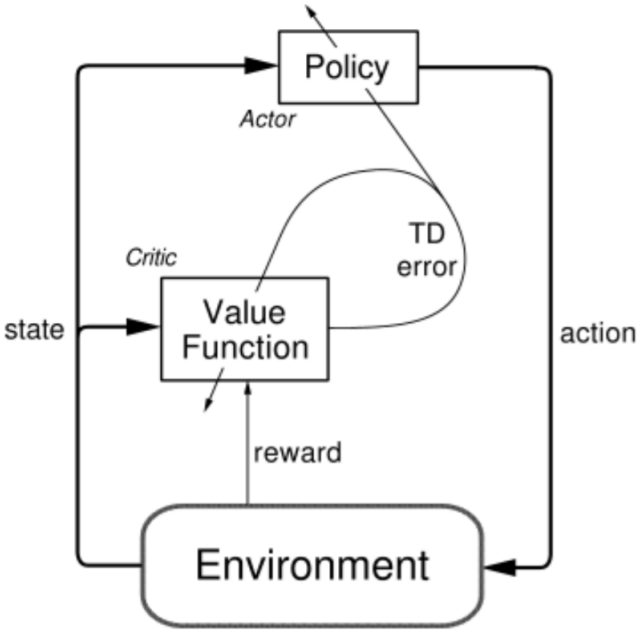


Figura 1.6: Esquema gráfico del algoritmo Actor - Critic

[11] El funcionamiento básico del algoritmo Actor - Critic se basa en dos conjuntos diferentes de neuronas de salida, el primero de ellos, relacionado con el Actor, se encarga de asociar una probabilidad a cada una de las posibles acciones que el agente puede tomar sobre el entorno. El segundo conjunto consiste en la parte del Critic, encargado de evaluar cómo de buena fue la acción tomada por el Actor. [10] A través de la Figura 1.6 se analiza el funcionamiento del algoritmo Actor - Critic. En función del estado actual y de la política del Actor (aquella que establece qué acción tomar), el Actor escoge una acción sobre el entorno provocando una transición a un nuevo estado y obteniendo una recompensa (reward) sobre la acción tomada ,siendo a la vez transferida junto con el estado nuevo, a la función valor del crítico que será la encargada de estimar cómo de buena fue la acción tomada por el Actor.

[6] Los tipos de RL se clasifican según los componentes principales del RL de los que se disponga en cada tipo de problema a resolver, entre los que se encuentran: Policy - based, Value - based y Model -based. Como se muestra en la Figura , el Algoritmo Actor - Critic se sitúa en el conjunto de Policy - based (Actor) y Value -based (Critic):

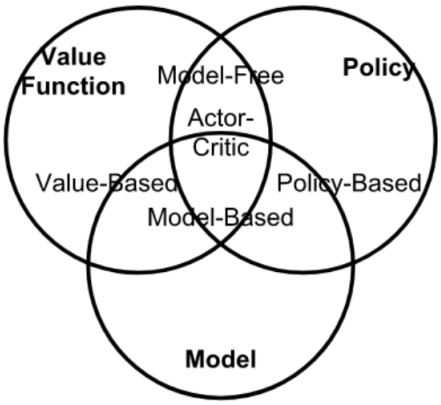


Figura 1.7: Esquema gráfico tipos de RL

En el caso de este proyecto, se ha optado por utilizar un algoritmo de RL por su semejanza con los problemas que se plantean resolver. Se dispone de un controlador de trayectorias MPC para un vehículo autónomo que es totalmente funcional, pero quiere ser optimizado para reducir los errores cinemáticos de distancia lateral y de yaw que actualmente tiene.

Las acciones que el agente tomará para transicionar de estados consistirán en modificar los pesos asociados a los errores de distancia lateral (ed) y error de yaw (ey) de la función de costes J del controlador MPC. Como el número de acciones a tomar para estimar unos resultados óptimos es bastante elevado, en vez de emplear el algoritmo DQN se optó por escoger el algoritmo Actor - Critic para así disponer de un conjunto de acciones ilimitadas por tomar sobre el entorno por el agente y encontrar así unos valores de pesos óptimos de distancia lateral y de yaw para la función de costes del controlador MPC.

Para definir desde un punto de vista gráfico en qué consisten los errores de distancia lateral y de yaw de un vehículo, ya que serán los estados sobre los que trabajará el algoritmo de aprendizaje neuronal en este proyecto, se procede a mostrar una breve descripción y dos esquemas gráficos en relación a estos errores.

- **Error de distancia lateral (ed)**

  Consiste en la longitud de la recta entre el centro de masas del vehículo y el eje norte-sur del vehículo con respecto al centro de la trayectoria deseada a seguir por el vehículo. Es muy importante que en un futuro vehículo autónomo este error sea lo más bajo posible, ya que un error grande de distancia lateral durante el seguimiento de una trayectoria podría provocar una colisión lateral con otros vehículos que estén circulando por la calzada, con peatones, ciclistas o con cualquier objeto presente en la vía pública. En la siguiente imagen [12] se muestra un esquema gráfico del error de distancia lateral de un vehículo:

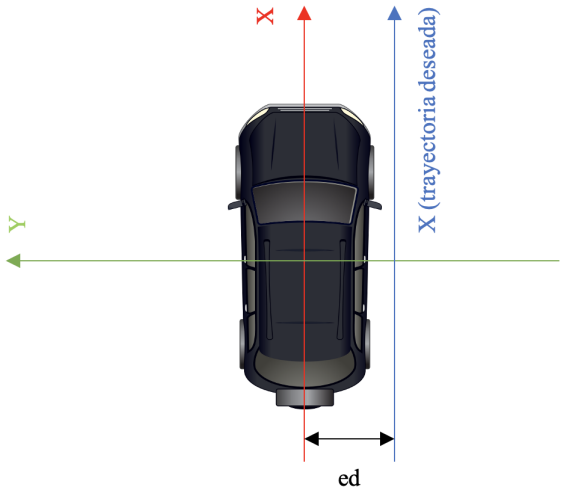


Figura 1.8: Error de distancia lateral en un vehículo

- **Error de yaw (ey)**

  El yaw corresponde con el ángulo de giro del vehículo respecto a su eje vertical. Define la orientación de la dirección del vehículo durante su trayectoria. Es importante también que su error sea lo más bajo posible para que el vehículo pueda tomar las curvas correctamente, aunque es más importante aún que sus variaciones sean mínimas para así garantizar una conducción agradable a

los pasajeros del vehículo, que de lo contrario, consistiría en repetidas oscilaciones del vehículo que pueden ser incómodas para el confort de los usuarios, así como perjudiciales para los componentes mecánicos del vehículo. En la siguiente imagen [12] se muestra un ejemplo gráfico del error de yaw de un vehículo:

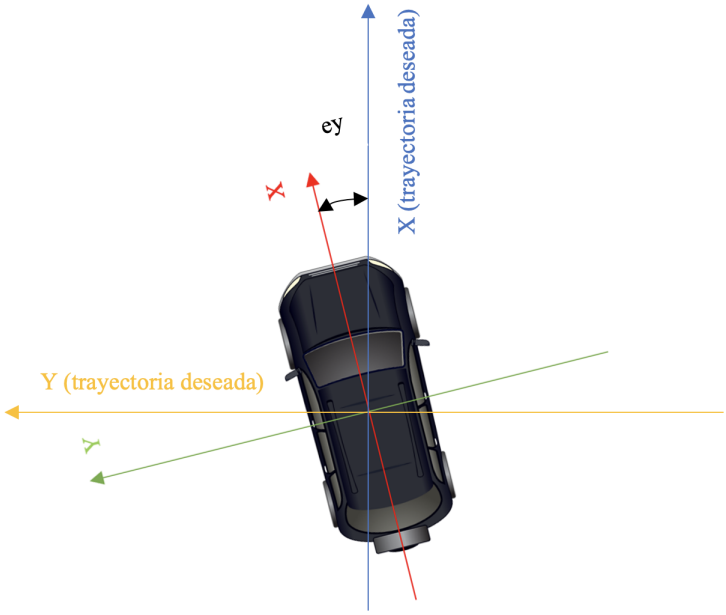


Figura 1.9: Error de yaw en un vehículo

Como anteriormente fue razonado, los errores de un controlador MPC dependen del valor que se le atribuye al peso *(w)* correspondiente en su función de costes *J*. Más en concreto, a la diferencia entre los pesos de esta función. Inicialmente en el diseño del MPC, se le atribuyeron unos valores de peso de distancia lateral de wd = 10.000 y peso de yaw de wy = 10.000 de forma experimental. Como punto de partida para estimar los pesos de distancia lateral y de yaw óptimos, se tomarán estos pesos previamente mencionados, pues con esos valores el vehículo es capaz de moverse a lo largo de toda la trayectoria de entrenamiento, aunque con cierto error que deberá ser reducido gracias a los pesos óptimos que serán estimados. Con el objetivo de encontrar los valores óptimos de estos pesos para así minimizar los errores de distancia lateral y de yaw, es por lo que se optó por utilizar este tipo de técnicas de aprendizaje por refuerzo para optimizar el funcionamiento del MPC.

Tras varios episodios de simulación, la arquitectura de red propuesta irá iterando distintos valores de pesos de distancia lateral y de yaw de forma independiente en función de sus correspondientes errores, obteniendo así una recompensa positiva si el error se ha reducido en cada iteración o nula en caso contrario. De esta forma se obtendrá el valor óptimo de estos valores y se reducirá el error de distancia lateral y orientación, mejorando así el comportamiento del controlador MPC y por lo tanto el seguimiento de la trayectoria por parte del vehículo autónomo. Los valores de pesos óptimos serán aquellos que consigan que el vehículo consiga moverse la mayor parte de la trayectoria de entrenamiento con las restricciones de errores máximos que serán impuestas. De esta forma se obtendrán unos pesos que posteriormente serán implementados en el controlador MPC y se observará cómo se reducen significativamente ambos errores respecto a su versión de partida.

Algoritmo Actor - Critic

Dentro de las técnicas de aprendizaje por refuerzo el algoritmo Actor - Critic fue escogido para resolver el problema planteado en este proyecto, pues posee un comportamiento idóneo para optimizar el controlador MPC.

[13] [14] [15] En este algoritmo el sistema parte de un estado inicial en el que el Actor escogerá una acción basada en una política correspondiente a una distribución de probabilidad entre las posibles acciones que dará lugar a la transición a un nuevo estado, provocando en consecuencia una recompensa a la acción tomada. El Critic se encarga de evaluar la política que ha empleado el Actor para escoger la acción. Asimilándolo a un problema de aprendizaje supervisado, el Actor se encarga de generar las salidas reales de la red y el Critic de crear la base de datos de las salidas deseadas de la red para así después proceder al entrenamiento de la misma.

Para comprender el funcionamiento del algoritmo Actor - Critic, se procede a mostrar su aplicación sobre el entorno CartPole, sintetizado de la web oficial de Keras [16].

CartPole

El entorno CartPole consiste en un entorno compuesto por carro con un poste unido a su centro que se desplaza a lo largo de una trayectoria horizontal sin fricción hacia derecha e izquierda. Las acciones que el agente puede tomar sobre el carro es aplicar una fuerza positiva o negativa, que se provoca un desplazamiento del carro hacia derecha o izquierda. El objetivo principal de la red es que el agente sea capaz de aplicar la acción correcta (+F o -F) para que el poste no se caiga del carro.

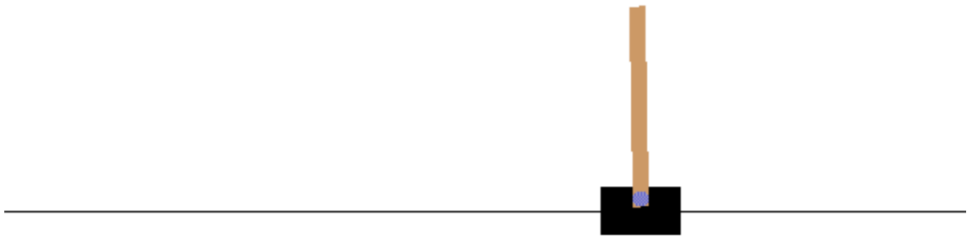


Figura 1.10: Entorno CartPole

Para entrenar la red se imponen unas restricciones que consten en la distancia del carro respecto al origen, para comprobar si el carro se ha salido de los límites laterales, con un valore máximo de $\pm$ 2.4 metros, y el ángulo máximo de inclinación del poste respecto a la vertical de su centro de masas, que corresponde con un ángulo de $\pm$ 15º.

La red es entrenada tomando como entrada el vector de estados compuesto por su posición, velocidad lineal, ángulo de inclinación del poste y su derivada. La arquitectura de la red actor se compone de cuatro neuronas de entrada, cada una correspondiente con cada una de las variables del vector de estado, con función de activación lineal. Después se propaga a través de una capa oculta de 128 neuronas con función de activación ReLU y por último a las tres neuronas de salida. Dos las mismas corresponden al Actor

y tienen función de activación softmax (valores entre 0 y 1) para así asignar un valor de probabilidad a cada una de las acciones posibles (aplicar una fuerza positiva o negativa). La otra neurona de salida corresponde con el Critic y posee función de activación lineal. El Critic es el encargado de estimar el valor de los rewards esperados en ese episodio (target de nuestra arquitectura de red neuronal). En la siguiente imagen se muestra un esquema de la arquitectura de red Actor-Critic para el caso del entorno CartPole:

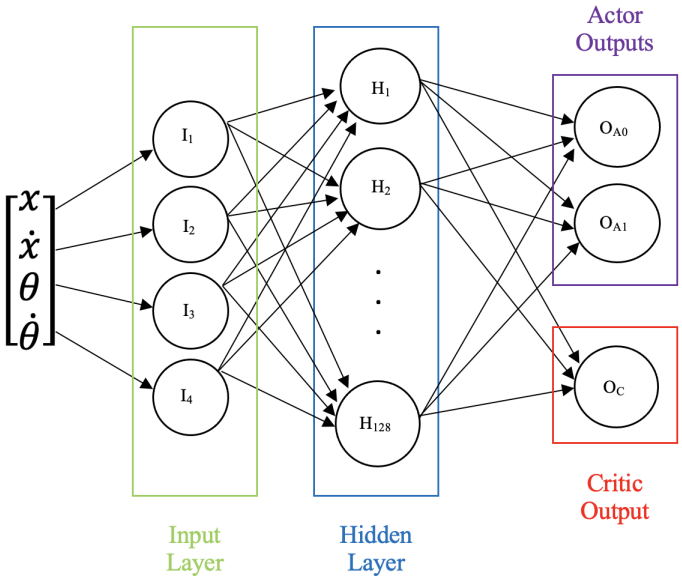


Figura 1.11: Arquitectura de red Actor-Critic en CartPole

Una vez la red ha estimado un valor de probabilidad asociado a cada acción, gracias a la función *np.random.choice* de la librería de Python *numpy*, se escoge una de las dos acciones atendiendo a los valores de probabilidad dados por el Actor. La acción tomada es enviada al entorno para realizar un desplazamiento hacia izquierda o derecha del carro dependiendo si la acción tomada es 0 (Fuerza negativa, desplazamiento hacia la izquierda), o 1 (Fuerza positiva, desplazamiento hacia la derecha).

Del entorno de CartPole se recibe el estado nuevo del carro, la recompensa asociada a la acción tomada (reward) y una variable booleana *done*. La política de rewards consiste en evaluar si el carro se ha salido de las restricciones impuestas. De no ser así, se premiará la acción tomada con un reward +1 y la variable done llevará asociada un valor booleano de False. De lo contrario, la variable done tomará un valor True, indicando así que el episodio actual debe finalizar ya que con la acción tomada se ha incumplido alguna de las restricciones, asignando además un valor de reward de 0 en esa iteración.

Una vez finaliza el episodio, se procede a entrenar la red calculando las funciones de pérdidas, haciendo el backpropagation y por último actualizando los pesos y biases de la red para el siguiente episodio de entrenamiento.

Entrenamiento de la red

Como había sido razonado anteriormente, a diferencia del aprendizaje supervisado, en el aprendizaje por refuerzo no existe una base de datos sobre la que comparar las salidas deseadas para la red con la salida real para así proceder al cálculo de la función de pérdidas y al backpropagation para entrenar la red neuronal actualizando sus pesos. [17] En este tipo de técnicas, el entrenamiento de la red se lleva a cabo mediante una política basada en los los rewards de cada iteración del episodio de entrenamiento. La red Critic irá creando una base de datos con las salidas deseadas (rewards esperados) para cada iteración del episodio de entrenamiento.

Basado en la técnica de aprendizaje por refuerzo Q-learning, esta política de rewards en el algoritmo Actor-Critic, pretende asignar un valor Q para cada par estado-acción que será función de los rewards obtenidos por iteración en un episodio de entrenamiento.

Las salidas reales de la red se obtienen aplicando una función que pretende dar más peso a los rewards más actuales que a los pasados, pues se tiene mayor conocimiento del entorno según avanzan las iteraciones. [17] Para cada iteración (k) por episodio, dicha función corresponde con la ecuación que se muestra a continuación:

$$Q(s_k, a_k) = r(s_k, a_k) + \gamma \cdot Q(s_{k+1}, a_{k+1}) \tag{1.5}$$

Esta expresión aproxima el valor Q a cada par estado acción, descontando las recompensas futuras por un factor γ entre 0 y 1 asignando así mayor valor a los rewards actuales cuanto más cerca de 1 se encuentre. Para este proyecto se optó por escoger un valor de $\gamma = 0.99$ para dar así mucho peso a las recompensas futuras de las que se tiene mayor conocimiento del entorno.

Una vez obtenidos los valores de salida reales de la red, estos serán normalizados restando a cada valor la media del total y dividiéndolo entre la desviación típica, para así obtener unos valores que puedan ser correctamente comparados con los valores estimados por el Critic, según la ecuación que se muestra a continuación:

$$Q(\hat{s_k}, a_k) = \frac{Q_{(s_k,a_k)} - \bar{Q_{(s_k,a_k)}}}{\sigma_{Q(s_k,a_k)}} \tag{1.6}$$

Una vez termina un episodio de entrenamiento, con la expresión anteriormente mencionada y con los valores estimados por al red Critic se procederá a calcular la función de pérdidas necesaria para entrenar la red y para así después hacer el backpropagation (descenso del gradiente) para actualizar los parámetros de la red para el siguiente episodio, consiguiendo así que la red aprenda de los resultados obtenidos.

De esta forma, para cada episodio de entrenamiento, se irán obteniendo una serie de rewards reales (value) vs esperados (target), que inicialmente sus gráficas no se asemejarán, pero que conforme vayan pasando los episodios de entrenamiento ambos irán adquiriendo valores similares y su diferencia (diff) estará en torno a 0. En la siguiente imagen se puede ver para el episodio 0 de entrenamiento de CartPole los valores de rewards reales (value) y esperados (target) así como si diferencia (diff):

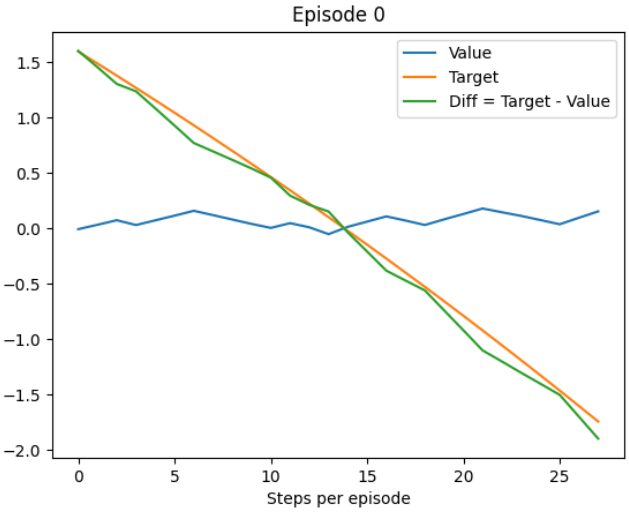


Figura 1.12: Rewards reales vs esperados episodio 0 CartPole

También se muestra a continuación la imagen con los valores de la función de pérdidas para el mismo episodio inicial:

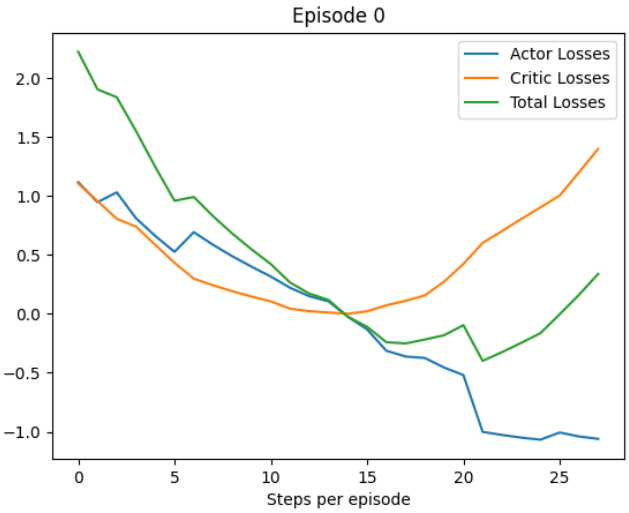


Figura 1.13: Valores de función de pérdidas episodio 0 CartPole

En cada episodio se van acumulando los rewards que corresponden con las veces que las acciones tomadas han sido buenas. En los primeros episodios, habrá muy pocos rewards acumulados ya que la red posee de muy pocas experiencias que conforme vayan avanzando los episodios, al tener más conocimientos de episodios pasados, los rewards acumulados irán siendo mayores. Este resultado puede apreciarse en la figura que se muestra a continuación:

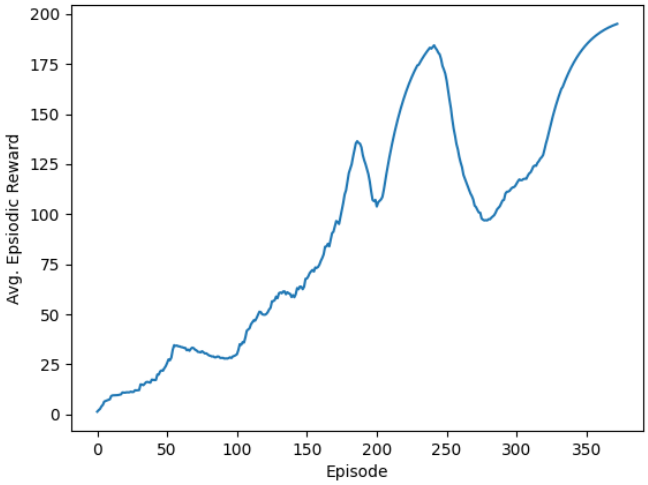


Figura 1.14: Rewards acumulados por episodio en un entrenamiento en CartPole

El problema se considerará resuelto cuando se llegue a un número de rewards acumulados igual a 195, que en este caso tuvo lugar en el episodio 370. Este valor corresponde con un parámetro de diseño de la red, que deberá ser escogido en función del número de iteraciones por episodio que se consideren las suficientes como para que el comportamiento sobre del agente sobre el entorno sea aceptable.

Es decir, en el caso del CartPole, en el episodio en el que se haya resuelto el problema, se habrán obtenido 195 rewards (iteraciones) en los que el carro no se habrá salido de ninguna de las restricciones impuestas, consiguiendo así tras varios episodios de entrenamiento que las acciones de control sean tomadas de la forma deseada. En la siguientes imagen se puede ver el gráfico de los rewards esperados (target) frente a reales (value) y su diferencia (diff) en el episodio en el que se consideró que la red había sido entrenada (370):

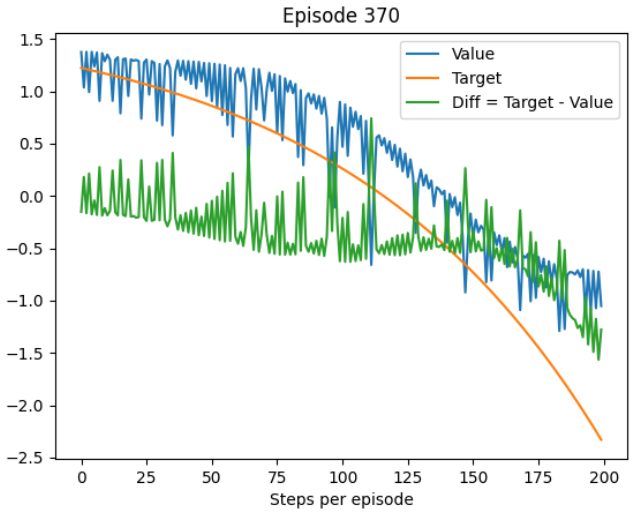


Figura 1.15: Rewards reales vs esperados episodio 370 CartPole

También se muestra a continuación el gráfico con los valores adquiridos por las funciones de pérdidas del episodio 370:

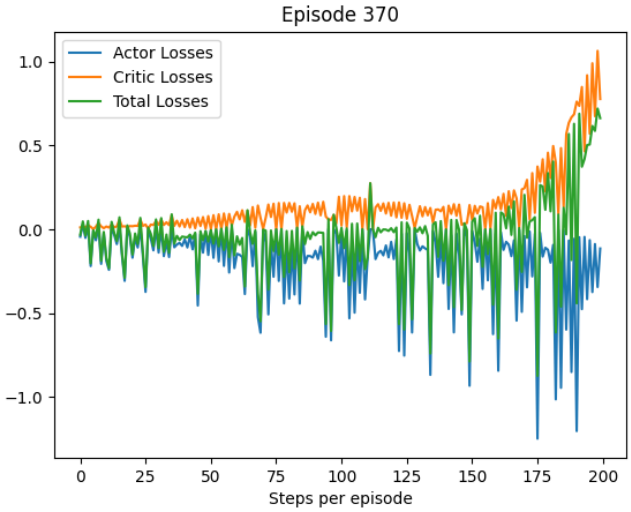


Figura 1.16: Valores de función de pérdidas episodio 370 CartPole

Se observa claramente cómo entre el episodio 0 y el 370 las gráficas definidas por value y target (Figuras 1.12 y 1.15 respectivamente), tienden a converger en valores similares obteniendo así una diferencia entre ambas (diff) con valores cercanos a 0 conforme avanza los episodios de entrenamiento. Además, las gráficas de las funciones de pérdidas, inicialmente (Figura 1.13) presentan valores mayores que en el episodio final, en el que se observa cómo para la mayor parte del episodio las pérdidas son prácticamente nulas (Figura 1.16) y por lo tanto la red está siendo entrenada correctamente.

En el caso del controlador MPC, para ser optimizado con el algoritmo Actor-Critic, el problema se considerará resuelto cuando se lleguen a un número de rewards en un episodio equivalentes a realizar la mayor parte de la trayectoria. Tanto para el caso del error de distancia lateral como el error de yaw, el entrenamiento se finalizará cuando se llegue a un episodio en el que, con los valores de pesos estimados para el MPC, el vehículo sea capaz de seguir la mayor parte de la trayectoria de entrenamiento sin salirse de las restricciones impuestas, que corresponden con los errores máximos que se desean que el vehículo alcance.

Al tratarse de un problema de optimización de valores (acciones que el agente MPC toma sobre el entorno vehículo), imponiendo restricciones (errores máximos), para proceder al aprendizaje al igual que en el caso del CartPole y disponer de un número ilimitado de acciones a tomar por el MPC (muchos valores de pesos), es por lo que se opta por escoger el algoritmo Actor-Critic en lugar de DQN para conseguir los objetivos de este proyecto, mediante los que se obtendrá un controlador MPC que controlará el movimiento de un vehículo autónomo a lo largo de una trayectoria con un error de distancia lateral y de yaw inferior con respecto a la versión inicial del controlador MPC.

A continuación se muestran dos gráficos correspondientes con el error de distancia lateral y de yaw de la versión inicial del MPC a lo largo de toda la trayectoria de entrenamiento:

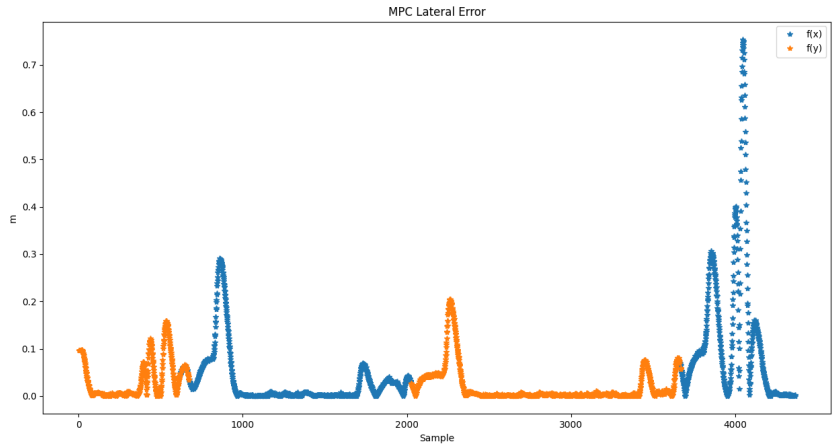


Figura 1.17: Error de distancia lateral del MPC original

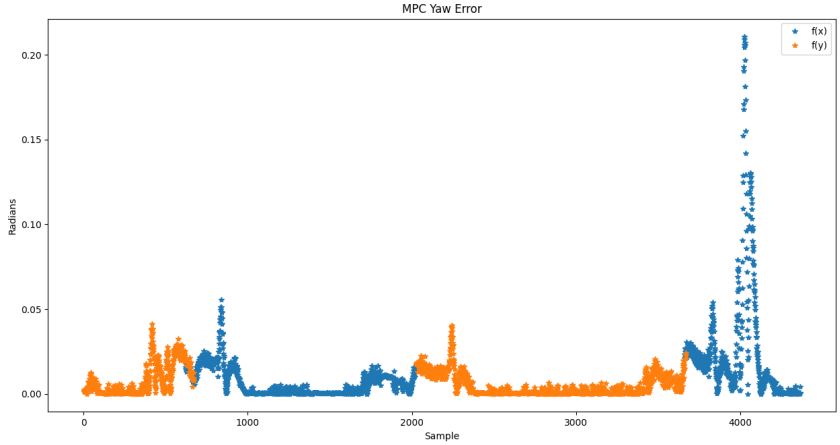


Figura 1.18: Error de yaw del MPC original - primera versión

Atendiendo a los errores descritos en las Figuras anteriores y con el objetivo de reducirlos lo máximo posible, para resolver este problema se impondrán unas restricciones consistentes en un error de distancia lateral (ed) inferior a 20 centímetros así como un error de yaw (ey) inferior a 0,07 radianes (4º sexagesimales aproximadamente). Estas restricciones serán las que indiquen cuándo un episodio de entrenamiento de la red debe finalizar, teniendo en cuenta que el episodio finalizará también en el caso de que el vehículo el en simulador colisione con algún objeto de la vía pública del circuito.

Capítulo 2

Estado del arte

2.1 Controladores MPC

Cuando un sistema de control adquiere cierta complejidad en su modelado y definición de la planta, es el momento en el que entran en juego los controladores MPC, que son capaces de satisfacer las limitaciones que poseen los controladores clásicos.

Con el avance tecnológico y la complejidad que van adquiriendo los sistemas debido a la dificultad del modelado de la planta y al elevado número de variables tanto de entrada como de salida a procesar por el controlador, la tendencia de los algoritmos de control [18], puede ser observada en el gráfico que se muestra a continuación:

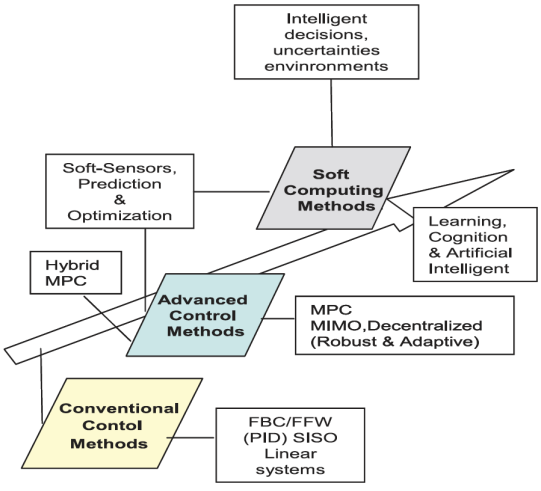


Figura 2.1: Desarrollo de sistemas de control

[18] En la Figura 2.1 se describe gráficamente cómo en los primeros años de la ingeniería de control se optaban por los métodos convencionales de control PID para sistemas lineales SISO (una única entrada y una única salida). Conforme fueron pasando los años y nacieron los sistemas digitales, era necesario controlar sistemas de múltiples entradas y salidas (MIMO) y que fueran adaptativos, en previsión a posibles cambios físicos en la planta del sistema a controlar. De ahí es de donde nacen los controladores MPC, capaces de controlar sistemas multivariables con plantas cuyo modelado puede llegar a ser muy complejo. Aunque el coste computacional de un controlador MPC es mucho mayor en comparación con

un PID o con un controlador óptimo LQR, los MPC ofrecen la ventaja de predecir acciones futuras de control con tan solo el conocimiento del sistema dinámico y el ajuste de los parámetros de su función de costes.

En el siguiente flujograma, [19] se muestra una clasificación de los controladores MPC.

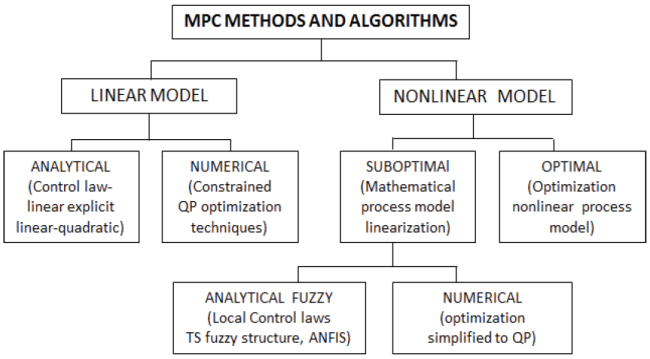


Figura 2.2: Clasificación de controladores MPC

La principal característica que clasifica a los MPC es su linealidad. Para los casos lineales, se puede distinguir entre analíticos y numéricos. Para los no lineales, está por un lado el caso de los óptimos empleados para optimizar procesos no lineales, y por otro lado los subóptimos que se dividen entre sí en análisis borroso y en numérico.

Los primeros controladores MPC se desarrollaron en la década de los años 70 para aplicaciones en procesos industriales en refinerías de petróleo e industrias petroquímicas. La principal ventaja que ofrece un controlador MPC y que tuvo tanto éxito en su desarrollo, es que es capaz de controlar sistemas multivariables con en los que se pueden imponer restricciones, mediante la optimización de una función de costes que presenta los errores cuadráticos de las variables a controlar, multiplicados por unos coeficientes independientes denominados "pesos"que atribuyen mayor o menor importancia a cada uno de los errores. El funcionamiento básico de un controlador MPC se puede observar en el diagrama de bloques de la Figura 2.3 [20]:

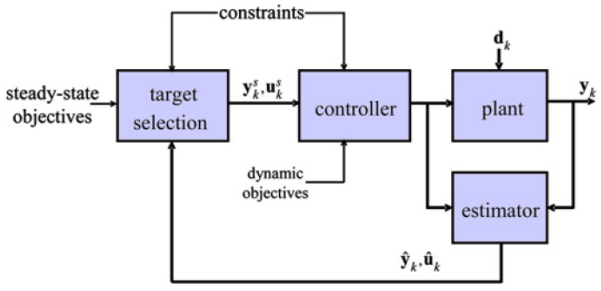


Figura 2.3: Diagrama de bloques de un MPC

Mediante la señal de referencia a seguir por el controlador y con las salidas reales del sistema, el MPC predice una salida futura para un horizonte temporal finito (número de muestras finitas), que después es comparada con la salida real del sistema, obteniendo así sus distintos errores, asociados todos en una misma ecuación de costes, que atribuyendo mayor o menor medida a cada uno de los errores con el valor de sus pesos, posteriormente se enviará al controlador optimizando así su comportamiento.

Es decir, en el caso de este proyecto, al controlador MPC se le pasa como señal de referencia la trayectoria que el vehículo debe seguir. Escogiendo un horizonte temporal de 3 para este caso, para cada muestra de control, el MPC generará 3 muestras futuras y se quedará con la que provoque un mejor comportamiento (la que produzca menor error) en el movimiento del vehículo sobre la trayectoria.

En este proyecto, como se expuso en la sección anterior 1 (Introducción), el objetivo principal es optimizar un controlador MPC mediante la búsqueda de los valores de los pesos óptimos de error de distancia lateral y de yaw con técnicas de redes DQN.

2.2 Redes DQN

El término DQN proviene de las siglas en inglés Deep Q Network, o redes profundas Q, cuyo fundamento se basa en emplear los algoritmos de redes neuronales artificiales sobre los algoritmos del aprendizaje Q.

Para entender el funcionamiento de las redes DQN, hay que entender el funcionamiento del Q-learning. El aprendizaje por refuerzo, consiste en que un agente sobre un entorno consiga provocar las mejores acciones de control sobre un entorno mediante una política de recompensas. El agente al provocar una transición hacia un nuevo estado al tomar una acción, es evaluado con un valor denominado Q, cuyo objetivo de este algoritmo es maximizar su valor, ya que, como fue mencionado antes, esto significaría un buen comportamiento sobre el entorno por parte del agente.

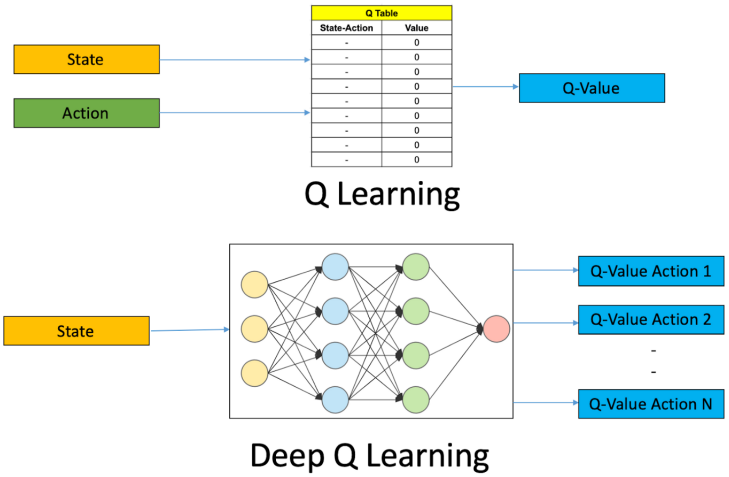


Figura 2.4: Diferencia entre Q-learning y DQN

Este algoritmo Q-learning se basa en generar una tabla bidimensional asociando un valor Q a cada par estado-acción del entorno sobre los que el agente puede transicionar. [21] Este algoritmo es muy útil y funcional para sistemas que no requieran un elevado número de estado y acciones, pero se complica cuando este comienza a ser mayor, ofreciendo una solución a este problema por parte de Mnih et al. [22] con el desarrollo de las redes DQN.

[22] El funcionamiento de las redes DQN se basa en emplear una arquitectura neuronal de redes fully-connected para generar los valores Q asociados a cada par estado-acción del Q-learning. En la Figura 2.4 [23], se puede observar gráficamente, como en el algoritmo DQN con tan solo la entrada del estado, la red neuronal estima los valores Q para las acciones, a diferencia del Q-learning donde se daban estos valores Q a cada par estado-acción.

El uso de algoritmos de aprendizaje por refuerzo en combinación con redes neuronales profundas está a la orden del día por sus excelentes resultados en diversos sectores como el industrial, marketing y

finanzas. Cabe destacar su empleo en operaciones de trading, cuyo esquema se muestra en la Figura 2.5 [24].

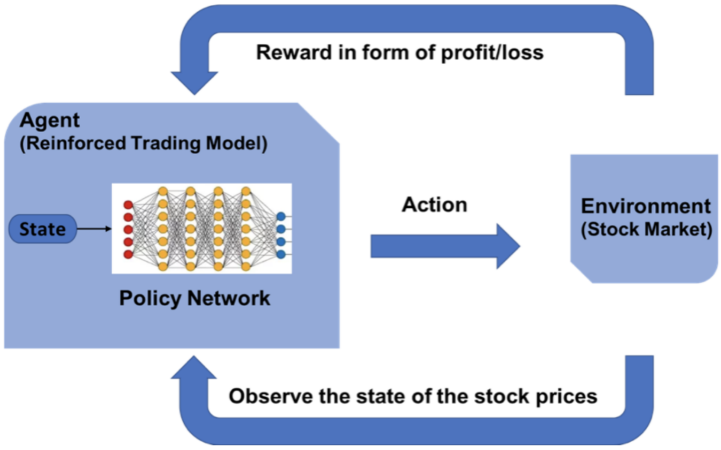


Figura 2.5: Aplicaciones de DQN a operaciones de trading

En este caso, se emplea una red DQN para estimar cuál debe ser la acción óptima a tomar sobre el entorno (mercado de valores) para obtener el mayor beneficio posible mediante su compra o venta en el momento oportuno. Las acciones tomadas por el agente, son recompensadas en caso de haber obtenido beneficio o no lo son en caso de haber obtenido pérdidas.

En este proyecto se optó por emplear un algoritmo de red DQN para optimizar los parámetros de los pesos de error de distancia lateral y de yaw del controlador MPC, debido a la gran cantidad de posibles acciones y estados que el controlador puede tener sobre el entorno.

El algoritmo Actor-Crítico, el cual también es un algoritmo de aprendizaje por refuerzo profundo, fue el escogido finalmente para este proyecto, por presentar mejores similitudes con los problemas a resolver como explicó en la sección anterior de Introducción 1, que principalmente se fundamenta en obtener el valor Q por medio de una función valor que evalúa cómo de buena fue la acción tomada por la parte del actor mediante una política, siendo ambos, tanto actor como crítico, neuronas de salida de toda la arquitectura neuronal planteada en el algoritmo (explicado con más detalle en la sección 4).

Capítulo 3

Entorno de trabajo

3.1 Hardware requerido

Para el desarrollo de este proyecto se utilizó un ordenador de sobremesa del laboratorio del grupo de investigación *INVETT* [2] con los componentes hardware necesarios para llevar a cabo las tareas de entrenamiento de redes neuronales en paralelo con la simulación en tiempo real de un vehículo autónomo sobre el entorno CARLA controlado por el MPC. Entre los componentes hardware más significativos de esta máquina se incluyen:

- Procesador: Intel Core i7-8700 CPU @ 3.20 GHz x 12
- Memoria RAM: 16 GB
- Tarjeta gráfica: Nvidia TITAN RTX

3.2 Software empleado

Tanto el **controlador MPC** previamente diseñado del que se parte en este proyecto, así como el algoritmo de aprendizaje neuronal ***Actor-Critic*** empleado para optimizar el comportamiento del MPC son desarrollados ambos en el lenguaje de programación **Python**.

Python es el lenguaje más empleado en el desarrollo de algoritmos de *Deep Learning* [25] ya que posee diversas librerías como *tensorflow* [26] con amplia variedad de clases orientadas al aprendizaje máquina así como la API de *Keras* [27] que incluye funcionalidades muy complejas como el cálculo de gradientes y optimización de redes que gracias a ella pueden ser implementadas de una forma bastante sencilla. Además, Python posee la ventaja de ser código abierto con la ventaja de no tener que estar al tanto de licencias. También dispone librerías de cálculo numérico y álgebra lineal como *numpy* que facilitan bastante el desarrollo software de operaciones matemáticas.

El sistema operativo empleado corresponde con una versión de *Linux*, en concreto con una la distribución de Ubuntu [28] (*Ubuntu 18.04.4 LTS*). La ventaja fundamental es que es un sistema operativo de software libre y puede ser configurado con la versión de Python que se empleará *(Python 3.7)* para el desarrollo y validación de los sistemas de control y aprendizaje sobre el vehículo autónomo de este proyecto.

3.2.1 Simulador CARLA

Para simular el control del MPC sobre el vehículo autónomo y observar cómo el algoritmo Actor-Critic actúa sobre el mismo,se emplea el simulador **CARLA(*CAR Learning to Act*)**.

[29] CARLA es un entorno de simulación de vehículos de código abierto que es capaz de procesar un entorno de conducción prácticamente real. [30]. El simulador puede ser configurado con una gran variedad de características presentes en la conducción real como pueden ser las condiciones climatológicas, aleatoriedad de peatones, situaciones de tráfico aleatorias, semáforos y diversos escenarios de carreteras entre otras muchas.

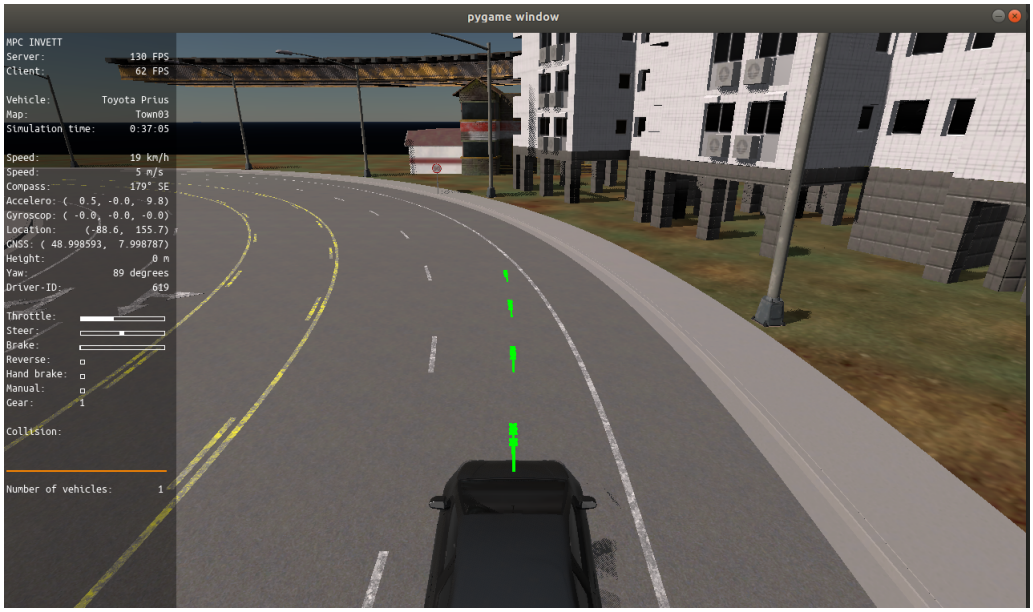


Figura 3.1: Controlador MPC sobre vehículo en simulador CARLA

Gracias a este simulador se podrá ejecutar en paralelo el controlador MPC para el control de la trayectoria previamente definida sobre un circuito urbano sobre el vehículo autónomo junto con el entrenamiento del algoritmo de aprendizaje, para así optimizar el MPC y además ir visualizando sus resultados sobre un entorno gráfico como se muestra en la Figura 3.1.

3.3 Trayectoria a seguir por el vehículo autónomo

Para validar el comportamiento del controlador MPC sobre el vehículo autónomo es necesario disponer de una referencia de la trayectoria deseada a seguir por el vehículo. Para la generación de dicha trayectoria se optó por crear un fichero de datos *json* que contiene las coordenadas discretizadas de la trayectoria de referencia.

Una descripción gráfica de la trayectoria es mostrada en la Figura 3.2 a continuación, mostrando la vista aérea o planta de la trayectoria de referencia, con las unidades de sus ejes expresadas en metros:

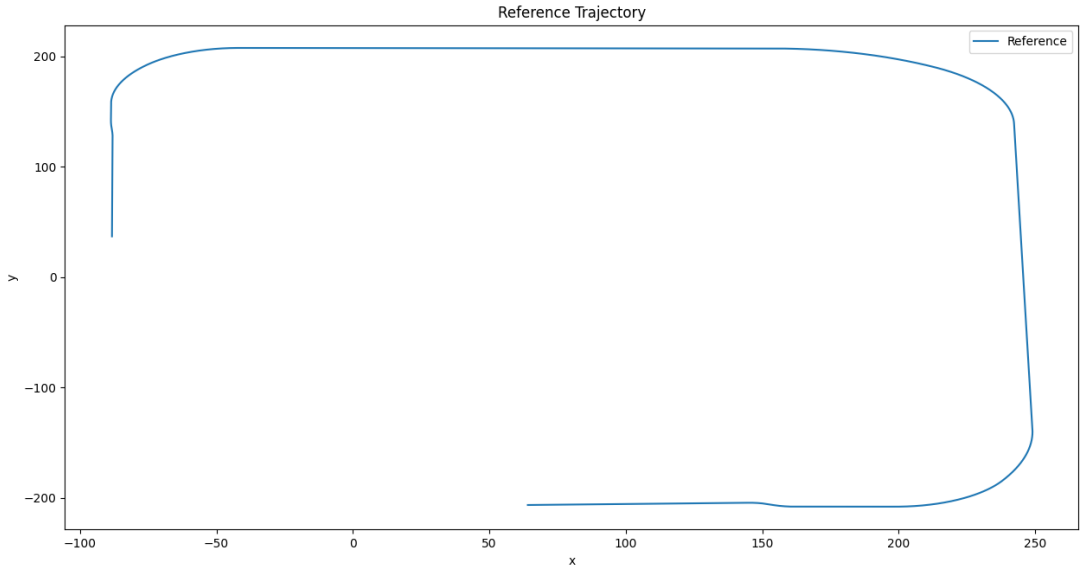


Figura 3.2: Trayectoria de referencia

Este será el escenario sobre el que controlará el vehículo autónomo con el MPC sobre el que se entrenará la red para así optimizar el funcionamiento del controlador y en consecuencia el comportamiento del movimiento del vehículo autónomo.

Capítulo 4

Propuesta de arquitectura neuronal para la optimización del MPC

Para optimizar el funcionamiento del controlador MPC con el objetivo de reducir los errores de distancia lateral (ed) y de yaw (ey) respecto a su versión original, la arquitectura neuronal basada en el algoritmo Actor-Critic corresponderá con el esquema gráfico que se muestra a continuación:

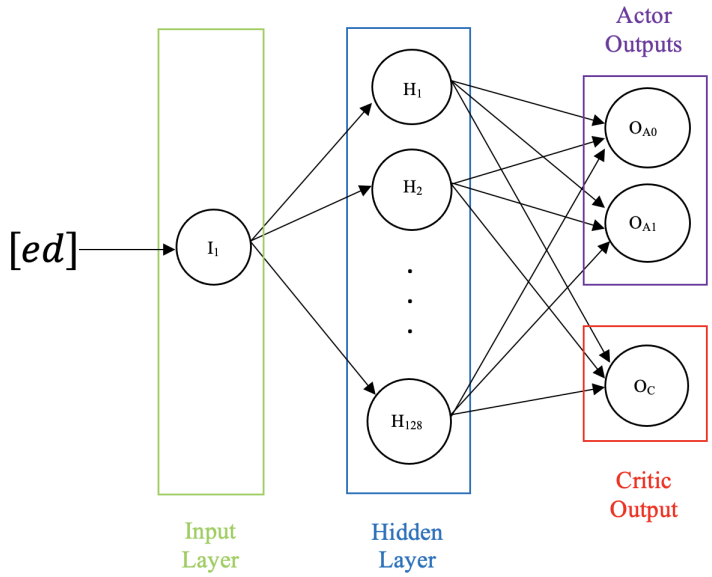


Figura 4.1: Arquitectura de red Actor-Critic para optimizar MPC

Tanto para obtener el valor del peso óptimo de error de distancia lateral (wd) como el del error de yaw (wy), el procedimiento que se empleará en ambos casos será el mismo. La arquitectura de la red será la mostrada en la Figura 4.1. Dicha red constará de una neurona de entrada con función de activación lineal, a la que se le pasará como entrada el valor del error de distancia lateral o de yaw (en función de qué peso del MPC se esté estimando, el otro permanecerá constante). Después, se propagará a una capa oculta compuesta por 128 neuronas con función de activación ReLU y por último a las tres neuronas de salida.

Dos de las neuronas de salida corresponden con el Actor, con funciones de activación softmax para asignar un valor de probabilidad a cada una de las posibles acciones, consistentes en incrementar o disminuir el mismo valor sobre el peso que se esté trabajando. La neurona de salida restante corresponderá con la del Critic, con función de activación lineal, que será la encargada de ir estimando el valor de los rewards esperados en cada iteración del episodio de entrenamiento, para así crear la base de datos sobre la que después proceder al aprendizaje y el entrenamiento de la red.

Basado en la arquitectura de red descrita anteriormente, el proceso de entrenamiento consistirá en el flujograma que se muestra a continuación:

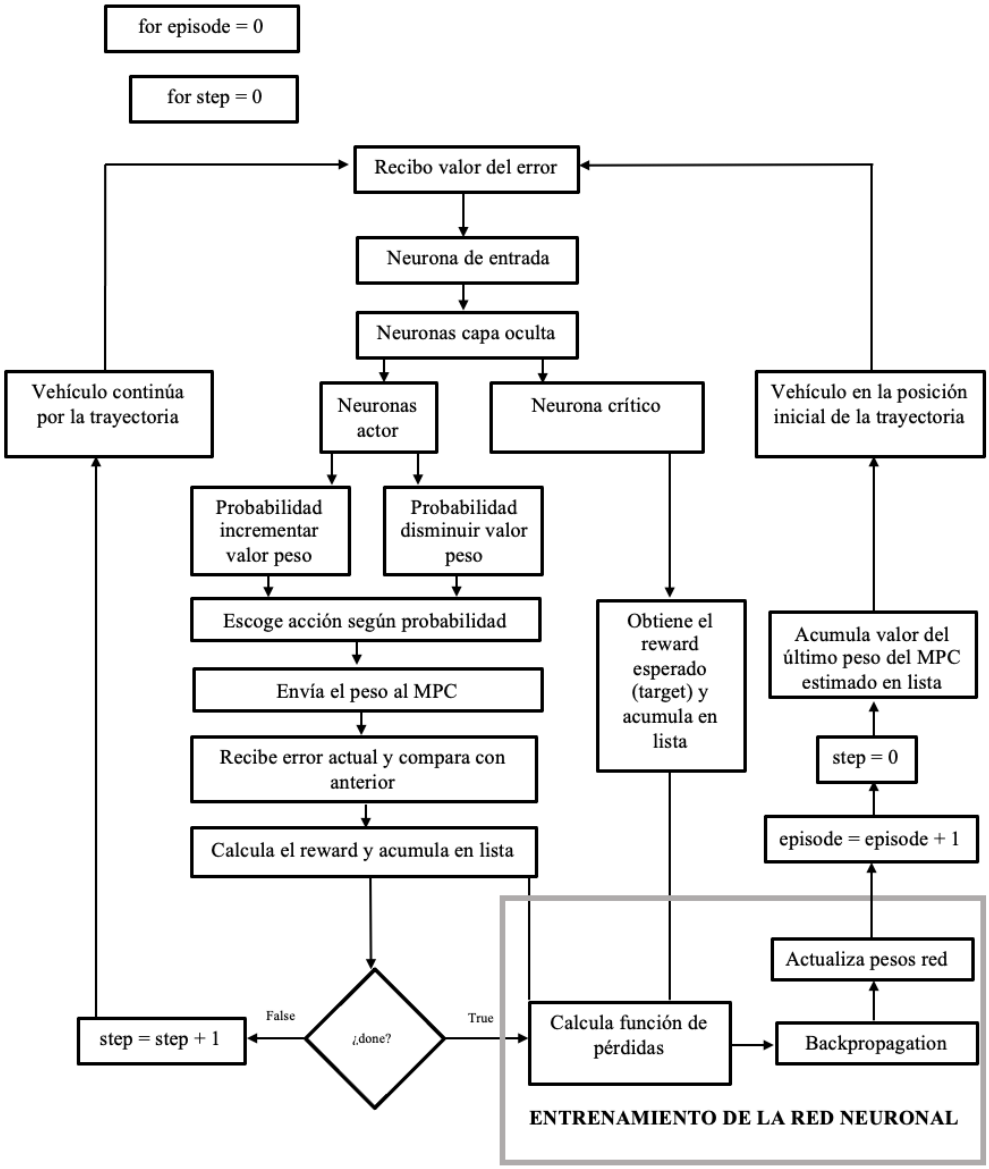


Figura 4.2: Flujograma de entrenamiento de la red neuronal

El funcionamiento del proceso descrito gráficamente en el flujograma será detallado a continuación para cada uno de los bloques.

En primer lugar, la red será entrenada hasta que se obtengan el número total de rewards por episodio máximos oportunos para que la red sea capaz provocar sobre el MPC un control del vehículo que consiga

recorrer la mayor distancia posible sobre la trayectoria de entrenamiento. Tanto el número de episodio (episode) como la iteración por episodio (step) son inicializados ambos a 0 para comenzar el proceso de entrenamiento.

Como entrada a la red, se recibe el valor del error a analizar para después ser enviado a las neuronas de la capa oculta y por último obtener las salidas del Actor y del Critic. Las salidas del Actor corresponden con la probabilidad asociada a incrementar o disminuir el valor del peso del MPC, acción que tras ser escogida se procederá a analizar se el cambio de error respecto a la iteración anterior obteniendo así el reward correspondiente (+1 si ha mejorado y nulo en caso contrario). Para el caso del Critic, se irá acumulando en una lista el valor del reward esperado para después ser comparado con el reward obtenido y así entrenar la red.

A continuación, se comprobará el valor de la variable booleana *done*. Esta variable tomará valor False si el valor del error del vehículo se encuentra dentro de los límites impuestos por la restricción o si el vehículo no ha colisionado con ningún objeto de la vía pública del simulador. Por el contrario, *done* tomará valor True si el error del vehículo está fuera de límites o si ha colisionado con algún elemento de la vía pública, indicando así que el episodio de entrenamiento debe finalizar y por tanto se debe proceder a entrenar la red neuronal calculando las funciones de pérdidas, haciendo el backpropagation y actualizando los pesos y biases de la red para así pasar al siguiente episodio de entrenamiento y mejorar continuamente sus predicciones.

Política de rewards

[17] Los rewards son la recompensa con la que se premian las acciones que han sido correctamente tomadas por la red en un algoritmo de aprendizaje por refuerzo. En este proyecto, la política de rewards consiste en premiar con una puntuación positiva (+1) cuando el error de la iteración actual sea inferior al de la iteración anterior, y con una puntuación nula en caso contrario. Esta política consiste en premiar aquellas acciones (valores de pesos del MPC estimados) que provocaron una disminución del error en estudio. El instante en el que el booleano done, que indica cuándo finaliza el episodio de entrenamiento, adquiere un valor True, se da un reward nulo, pues el vehículo incumplió alguna de las restricciones y por lo tanto el episodio finalizó.

En cada episodio de entrenamiento, se irán guardando el valor de los rewards para cada iteración para después calcular las funciones de pérdidas con su previa normalización como fue descrito en el capítulo de introducción en la sección del algoritmo Actor-Critic.

Funciones de pérdidas

[17] La función de pérdidas total del algoritmo Actor-Critic corresponde con la suma de la función de pérdidas del Actor y la función de pérdidas del Critic:

$$L = L_{Actor} + L_{Critic} \tag{4.1}$$

La función de pérdidas del Actor se obtiene calculando el logaritmo negativo de la diferencia entre los valores de recompensas estimados por la red crítica por iteración menos el valor Q estimado y normalizado

calculado con el descuento γ como se explicó en la introducción, en la sección de Aprendizaje por Refuerzo 1.3, que corresponde con la ecuación [9] que se muestra a continuación:

$$Q(s_k, a_k) = r(s_k, a_k) + \gamma \cdot Q(s_{k+1}, a_{k+1}) \tag{4.2}$$

Siendo *target* los valores esperados por la red (obtenidos por Critic) y *value* los reales obtenidos con la ecuación 4.2, se procede a aplicar la función de pérdidas del Actor como sigue:

$$L_{Actor} = -\log(value - target) \tag{4.3}$$

Para la función de pérdidas del Critic se emplean los mismos datos que en la función de pérdidas del Actor pero en este caso aplicando el algoritmo Huber por ser menos sensible que la función del error cuadrático medio.[17]

Backpropagation

La actualización de los pesos de una red neuronal se obtiene mediante el descenso del gradiente o backpropagation, de acuerdo a la ecuación 4.4 que se muestra a continuación:

$$w_{i_k} = w_{i_{k-1}} - \delta \cdot \frac{\partial L}{\partial w_i} \tag{4.4}$$

Donde el peso i-ésimo es actualizado para cada muestra k de la función de pérdidas en función de la muestra anterior menos el producto de la tasa de aprendizaje o *Learning Rate* (δ) multiplicado por la derivada parcial de la función de pérdidas *(L)* con respecto al peso en cuestión. De la teoría de redes neuronales artificiales, se conoce que para que la red aprenda, es decir, que se obtenga una convergencia entre los datos deseados y los estimados por la red, es necesario que el *Learning rate* (δ) se encuentre en valores de órdenes de magnitud negativos. Para este proyecto se utilizaron dos valores de *Learning Rate* distintos por mostrar resultados favorables para cada uno de los casos en cuestión para el entrenamiento de la red del algoritmo Actor - Critic.

Gracias a las librerías de Tensorflow y Keras, es bastante sencillo calcular el gradiente y actualizar los parámetros de la red neuronal. [26][27] El gradiente se obtiene con la clase *GradientTape* de Keras, con el método *gradient*, pasándole como argumentos cada una de las muestras de la función de pérdidas total y el método de la clase *trainable variables* del modelo de la red neuronal previamente configurado que contiene el valor de los pesos de la arquitectura neuronal.

Por último, los parámetros de la red neuronal se optimizan con *Adam* y se actualizan también gracias al método *apply gradients* de la clase optimizers de Keras, al que se le pasa como argumento el valor de *learning rate* que se considere óptimo para obtener una convergencia en el proceso de aprendizaje.

4.1 Optimización del peso de error de distancia lateral

En la primera versión del controlador MPC, el error lateral máximo que se obtenía en la trayectoria de entrenamiento tomaba valores de hasta 70 centímetros en la curva más pronunciada que tenía lugar en el final de la trayectoria como pudo verse en la Figura 1.17.

Con el objetivo de reducir el error de distancia lateral a lo largo de toda la trayectoria lo máximo posible, era necesario establecer una restricción de error lateral sobre la que el algoritmo de aprendizaje por refuerzo tomara de referencia como punto final del episodio de entrenamiento de la red.

Como en la primera curva de la trayectoria el error lateral máximo era de 20 centímetros, este fue el valor escogido como restricción para el entrenamiento. Con esta restricción de 20 centímetros nunca se iba a conseguir que el vehículo recorriera toda la trayectoria, pero si que fuera entrenada un número de episodios suficientes como para obtener unos valores de pesos de distancia lateral óptimos que consiguiesen reducir el error en su mayor parte de la trayectoria en comparación con la versión inicial del MPC.

En su versión inicial, el controlador MPC tenía un valor de peso de distancia lateral, *en adelante* ***wd***, de wd = 10000. Este valor y otros cercanos (como wd = 12000 para el primer resultado), fueron tomados como punto de partida del entrenamiento de la red para así estimar un valor óptimo que consiguiera reducir el error de distancia lateral.

La arquitectura de red empleada para todos los aprendizajes del peso de distancia lateral fue descrita en el esquema de la Figura 4.1, en la que se pasa como dato a la neurona de entrada el error de distancia lateral con función de activación lineal, después a las 128 neuronas de capa oculta con función de activación ReLU y por último se obtienen en las neuronas de salida del actor la probabilidad (función de activación softmax) de incrementar o reducir el valor de peso de distancia lateral y en la neurona del crítico (función de activación lineal) el valor de salida deseado dado por la política de rewards como fue explicado en la sección de Aprendizaje por Refuerzo 1.3.

Primera versión de optimización de wd

Para la versión inicial del MPC se encontró un valor óptimo de wd que fue capaz de reducir el error de distancia lateral a lo largo de la trayectoria en comparación con la versión inicial. Los parámetros correspondientes a este entrenamiento se muestran en la Tabla 4.1 a continuación:

Parámetro	**Valor**
Learning Rate	0,001
wd inicial	12000
Variación de wd	±1
Tiempo de entrenamiento	2 horas

Tabla 4.1: Parámetros de entrenamiento primera versión wd

Tras 2 horas de entrenamiento los rewards acumulados por episodio de entrenamiento que se obtuvieron corresponden con la imagen que se muestra a continuación:

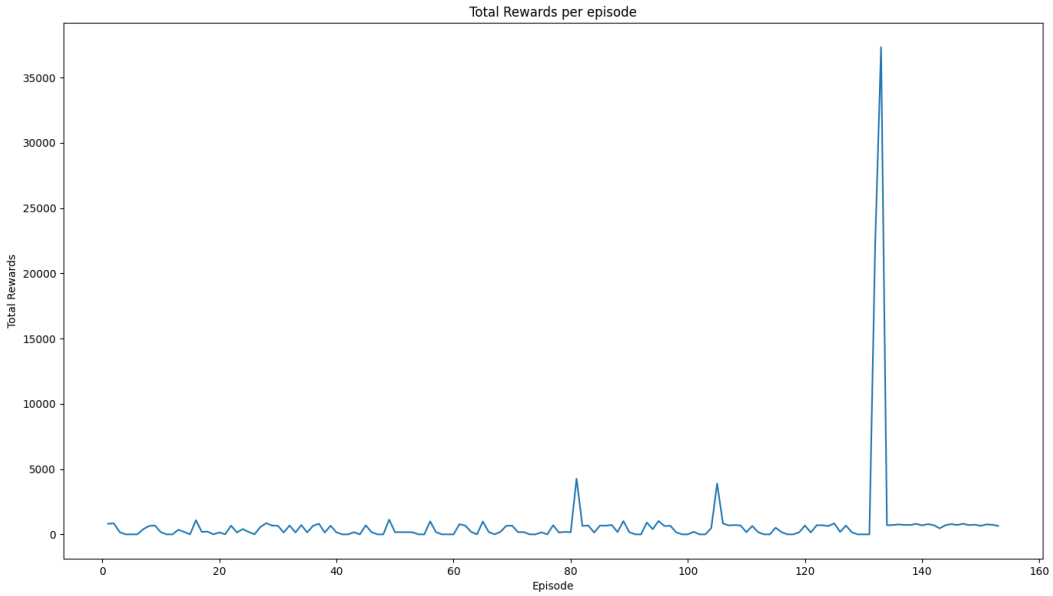


Figura 4.3: Rewards por episodio - primera versión optimización wd

El objetivo se alcanza en el episodio 131 que es en el que se obtiene un mayor número de rewards y por lo tanto el comportamiento del vehículo respecto al error de distancia lateral fue el mejor en comparación con el resto de episodios.

En este episodio, el vehículo fue capaz de seguir la mayor parte de la trayectoria dentro de la restricción de error lateral máxima de 20 centímetros. Para saber cuál es el valor wd óptimo de este episodio, se optó por quedarse con el valor final de este peso para así después ejecutar el MPC con dicho valor y analizar si el error de distancia lateral obtenido era menor con este nuevo valor de wd. En la siguiente imagen se muestra una gráfica con los valores finales de wd por episodio de entrenamiento:

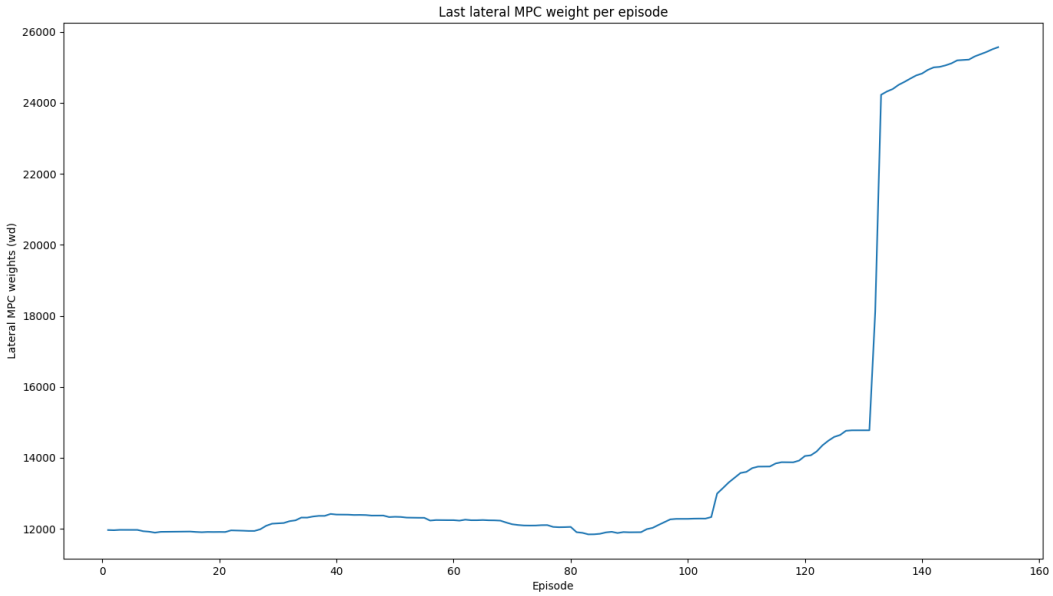
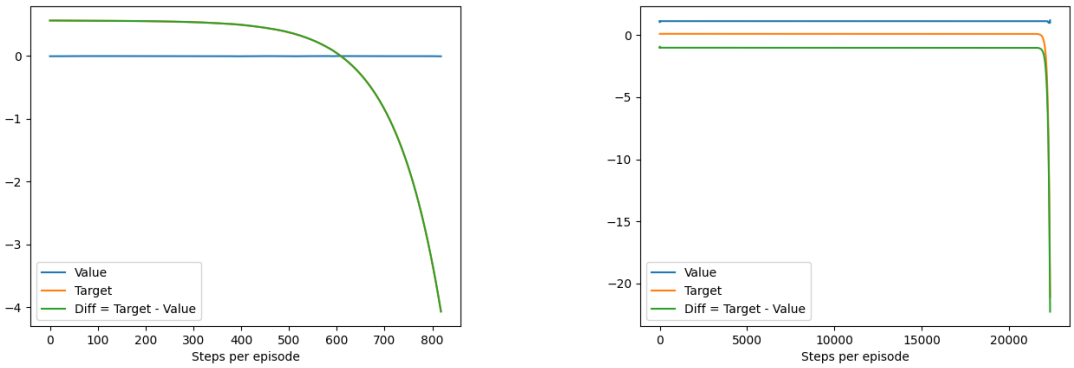


Figura 4.4: Valor final de wd por episodio - primera versión optimización wd

Atendiendo al máximo de la gráfica de rewards totales por episodio y de los valores finales de wd por episodio (Figuras 4.3 y 4.4 respectivamente), se puede observar cuál fue el rango en el que se encontró

el peso de distancia lateral de ese episodio. Con mayor exactitud se extrajo dicho valor del fichero *json* donde eran almacenados todos los datos, obteniendo así un valor de wd = 24255.

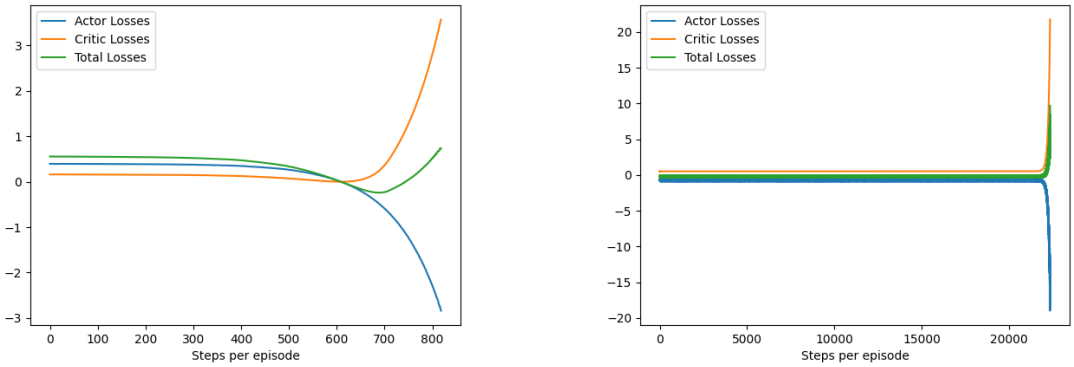
Para comprobar el funcionamiento de la arquitectura neuronal durante este proceso de aprendizaje, se proceden a mostrar los valores de salida deseados y reales de la red así como las funciones de pérdidas del episodio iniciales y del episodio en el que se dio lugar el valor óptimo, que corresponden con el 0 y 131 respectivamente:



(a) Episodio 0 (b) Episodio 131

Figura 4.5: Valores de salida reales vs esperados - primera versión optimización wd

Se observa cómo en el episodio inicial (Figura 4.5a) la diferencia entre los valores de salida deseados y reales es elevada así como su función de pérdidas. En el episodio 131 (Figura 4.5b) se aprecia claramente cómo los valores de salida (azul) se asemejan a los deseados (naranja) llegando así a una convergencia resultando en unos valores de función de pérdidas prácticamente nulos para este episodio (Figura 4.6b).



(a) Episodio 0 (b) Episodio 131

Figura 4.6: Función de pérdidas - primera versión optimización wd

En el episodio 131 el valor de peso de distancia lateral final fue de wd = 24225. Este valor será considerado como peso óptimo de esta primera versión del controlador MPC que será ejecutado de nuevo a lo largo de toda la trayectoria para así después comparar los valores de error de distancia lateral de la versión con wd = 12000 (valor de wd de partida del entrenamiento que fue diseñado inicialmente de forma experimental en el MPC con un funcionamiento similar a wd = 10000). Los resultados obtenidos se muestran en la imagen a continuación:

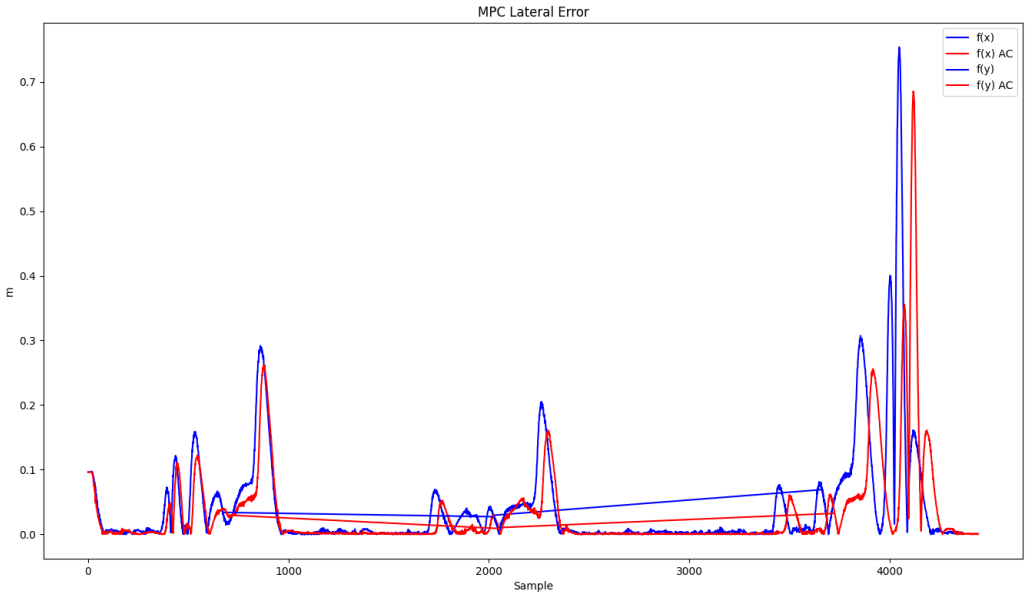


Figura 4.7: Comparación de error de distancia lateral tras optimización- primera versión

En color azul se muestra el error de distancia lateral a lo largo de la trayectoria del MPC inicial con un wd = 12000. En rojo se muestra el mismo error pero en este caso con el peso wd = 24225 estimado por la arquitectura Actor-Critic. **Se observa claramente cómo este error fue reducido con el valor de peso estimado por el aprendizaje neuronal y por lo tanto queda validado y aceptado el comportamiento de la arquitectura Actor-Critic empleada como primer resultado de este proyecto.**

Segunda versión de optimización de wd

La arquitectura del controlador MPC fue modificada en algunos aspectos mientras se desarrollaba este proyecto. En la nueva versión del MPC, los pesos de error de distancia lateral y de yaw obtenidos de forma experimental se situaron ambos en valores de 1, haciendo así que hubiese que adaptar el sistema de aprendizaje de este proyecto a la nueva versión del controlador.

Con el algoritmo Actor-Critic se procede a optimizar de nuevo el peso asociado al error de distancia lateral buscando un valor del mismo que consiga reducirlo en mayor parte a lo largo de la trayectoria de entrenamiento.

Para adaptar el sistema de aprendizaje a la nueva versión del controlador MPC hubo que realizar una serie de cambios en los parámetros del mismo. Como en esta nueva versión, el valor de wd experimental se situaba en 1, ahora las variaciones de incrementarlo o disminuirlo hubo que establecerlas en $\pm$0,01. La tasa de aprendizaje fue incrementada en un orden de magnitud respecto a la versión anterior por mostrar mejores resultados que los anteriores, siendo para este caso el *Learning Rate* = 0,01.

Uno de los problemas principales encontrados en el proceso de aprendizaje fue el largo tiempo de entrenamiento para obtener unos resultados válidos. Para reducir el tiempo de aprendizaje se propusieron y llevaron a cabo dos soluciones:

- **Empleo de la GPU para el aprendizaje máquina.** Se comprobó que los procesos del cálculo de la función de pérdidas llegaban a durar entorno a 2 minutos por episodio de entrenamiento, por lo que se optó por implementar la GPU del equipo para llevar a cabo este tipo de operaciones, consiguiendo así mayor rapidez en el entrenamiento de la red neuronal.

- **Ampliar el tiempo de muestreo.** Para cada episodio de entrenamiento, se tenían entre 25.000 y 30.000 muestras sobre las que realizar cálculos. Se optó por reducir el número de muestras por episodio ampliando el tiempo de muestreo del proceso de aprendizaje, y se comprobó que se obtenían mejores resultados y de forma más eficiente respecto a la versión anterior, pues no se demoraba en exceso el proceso de aprendizaje, y con un periodo de muestro de 100 ms, era suficiente para obtener entorno a 300 muestras por episodio, suficientes para que la red aprendiera con unos resultados favorables.

En esta nueva versión del controlador MPC el error lateral a lo largo de la trayectoria varía respecto a su versión original, mostrado en la Figura 1.17. Aunque el error de distancia lateral sea distinto en esta nueva versión respecto a la primera, el dato más importante por analizar consiste en el error máximo alcanzado en la trayectoria para establecer así la nueva restricción sobre la que el sistema de aprendizaje neuronal tome de referencia como fin de episodio. De los resultados del nuevo error lateral del MPC sin optimizar (wd = 1 e wy = 1) mostrados en la Figura 4.8, se puede obtener que el error de distancia lateral máximo se da entre las muestras 40 y 60 con un valor superior a los 20 centímetros. Por lo tanto, se tomará como restricción para el entrenamiento, al igual que en el caso anterior, un error de distancia lateral máximo de 20 centímetros para que así el sistema de aprendizaje neuronal estime un valor de wd óptimo que mejore el comportamiento del controlador sobre la trayectoria.

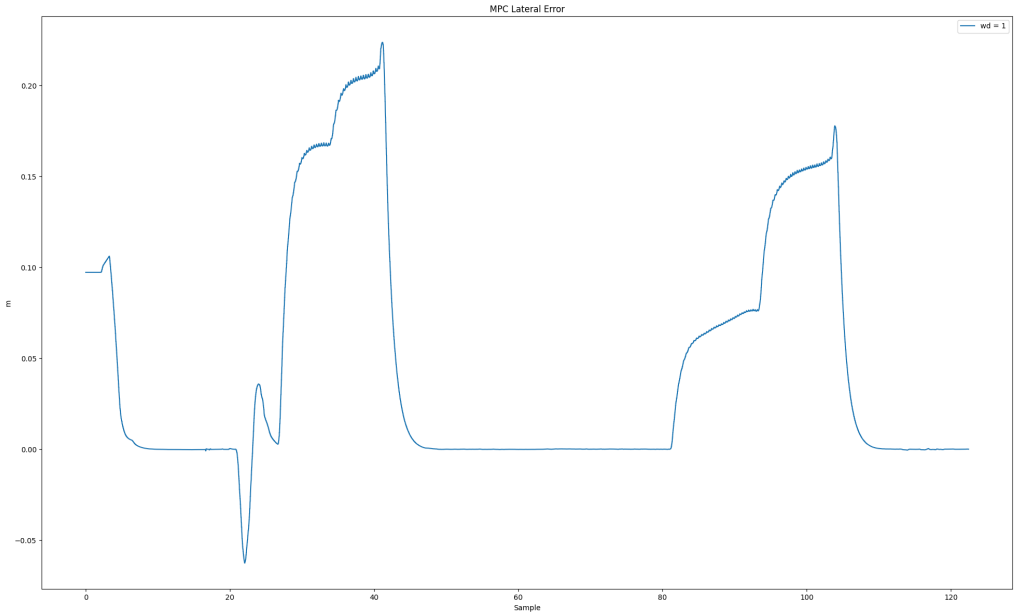


Figura 4.8: Error de distancia lateral del MPC original - segunda versión

Además, en esta segunda versión del controlador MPC, surge una nueva propuesta de encontrar un valor óptimo para los pesos de error de distancia lateral y de yaw. Con el objetivo de validar el funcionamiento del sistema de aprendizaje y obtener un mayor rango de valores de pesos, se optó por iniciar el entrenamiento del sistema desde un valor de peso negativo que provocara que el coche no fuese capaz de moverse a lo largo de la trayectoria y observar así cómo el sistema de aprendizaje conseguía evolucionar hacia valores de pesos positivos, que a su vez mejorarían el comportamiento del controlador MPC reduciendo en la mayor parte de la trayectoria el error en estudio en comparación con su versión inicial.

También se propuso una nueva idea en el método empleado en el caso anterior de encontrar un valor único de peso óptimo. En este caso, para disponer de una mayor variedad de valores sobre los que validar el sistema, se optó por encontrar en lugar de un valor concreto, un rango de valores de pesos y posteriormente validarlos comparando los resultados con los de la versión inicial de este segundo MPC (en el que el peso de error de distancia lateral es de wd = 1 y el peso de error de yaw, *wy en adelante* es igualmente wy = 1.

Con todas las modificaciones expuestas, se procede a entrenar de nuevo el sistema con los parámetros de la Tabla 4.2 que se muestra a continuación:

Parámetro	**Valor**
Learning Rate	0,01
wd inicial	-10
Variación de wd	±0,01
Tiempo de entrenamiento	3 horas

Tabla 4.2: Parámetros de entrenamiento segunda versión wd

Aunque el tiempo de entrenamiento en este caso es mayor que en la primera versión (1 hora más respecto a la anterior), comprobando los resultados obtenidos se observa que el mejor comportamiento del sistema no se da en los últimos episodios, por lo que podría haber sido detenido antes obteniendo también unos resultados válidos. Este hecho puede comprobarse en la gráfica de los rewards acumulados por episodio que se muestra a continuación:

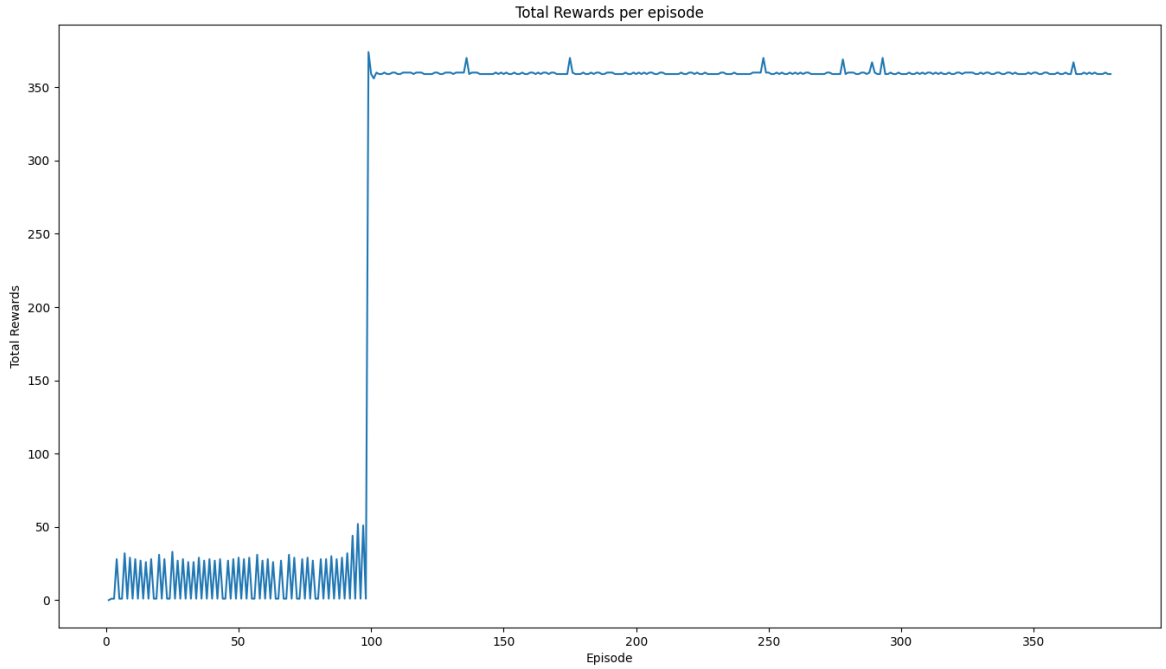


Figura 4.9: Rewards por episodio - segunda versión optimización wd

Cabe destacar cómo a partir del episodio 100 el vehículo es capaz de seguir prácticamente la mayor parte de la trayectoria de entrenamiento llegando a una distancia ligeramente mayor en algunos episodios (donde se observan picos superiores a 350 rewards) pero se observa que el sistema ya ha saturado y no es capaz de optimizar más su comportamiento.

Para obtener el rango de pesos de error de distancia lateral que provoquen un mejor comportamiento del MPC, en esta segunda versión durante el entrenamiento se van acumulando en un archivo *json* los valores de wd inicial y final por episodio para después calcular su diferencia y observar así en cuál de los episodios válidos (a partir del 100 en este caso) la diferencia entre ambos es mínima, pues esto significará que la variación de pesos por parte del sistema de aprendizaje ha sido baja, llegando así a una convergencia de valores de wd.

En la gráfica que se muestra a continuación se representa esta diferencia de wd inicial y final por episodio de entrenamiento:

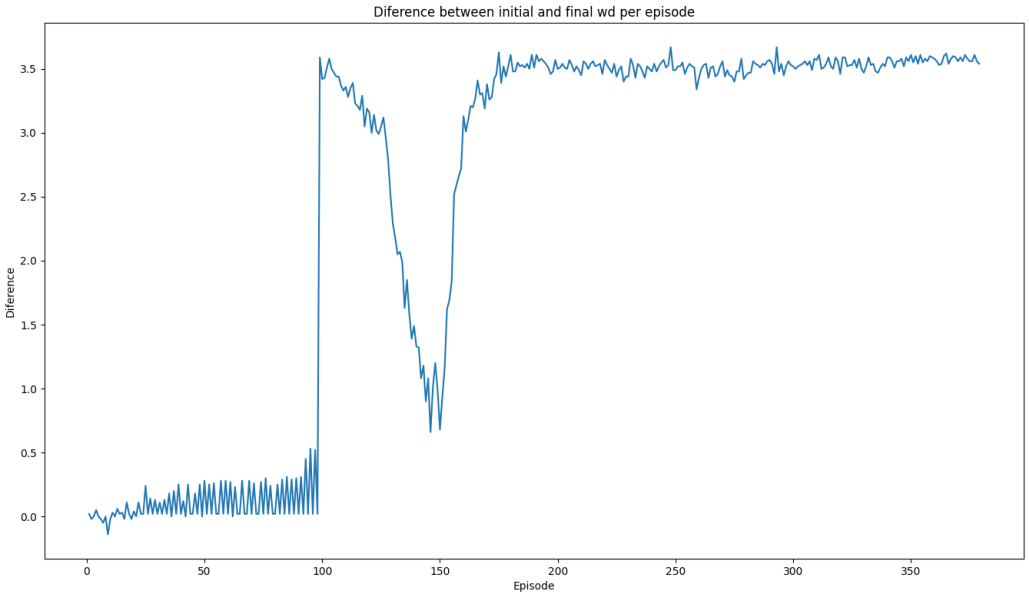


Figura 4.10: Diferencia entre wd inicial y final por episodio

Atendiendo a los valores de diferencia de wd inicial y final mostrados en la Figura 4.10 podrán considerarse válidos aquellos a partir del episodio 100 como fue razonado anteriormente. Se observa cómo en dos episodios entorno al 150 se obtiene la menor diferencia entre ambos valores. Del fichero *json* que almacena estos datos se extrae que estos episodios corresponden con el 146 y el 150.

En los episodios 146 y 150 son aquellos en los que la diferencia entre el valor de wd inicial y final es menor. Es decir, en estos episodios, la variación de los mismos habrá sido mínima, considerando así válidos los valores de wd dentro del rango de cada episodio. En la siguiente imagen se muestra una ampliación entre del rango de valores de wd entre los episodios 146 y 150:

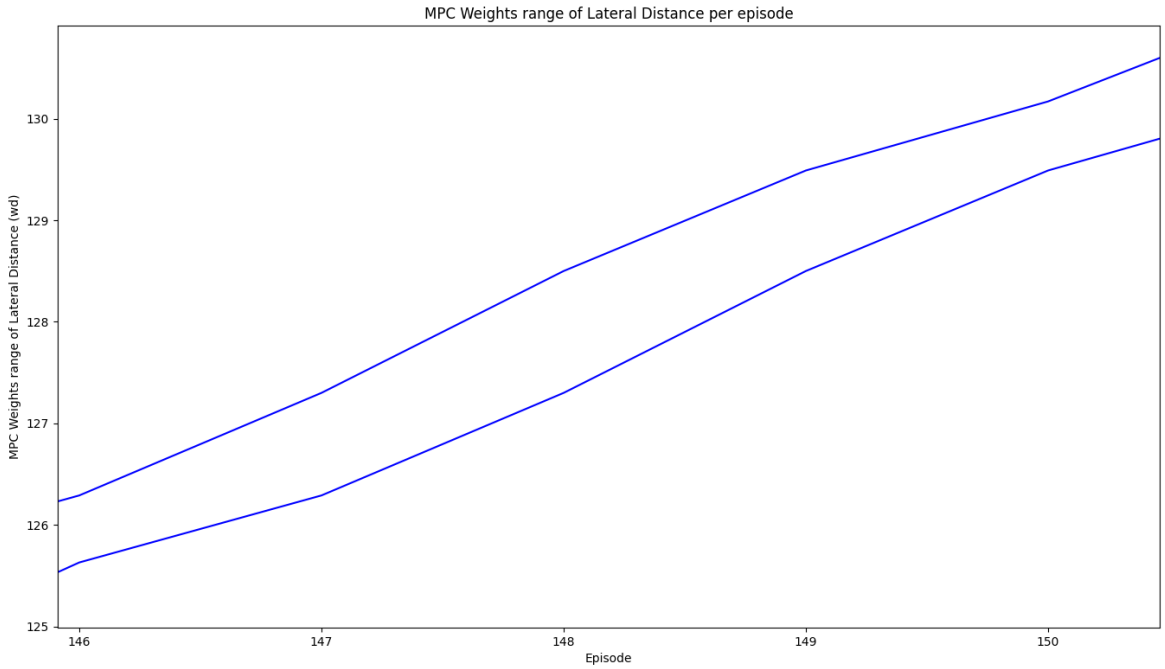
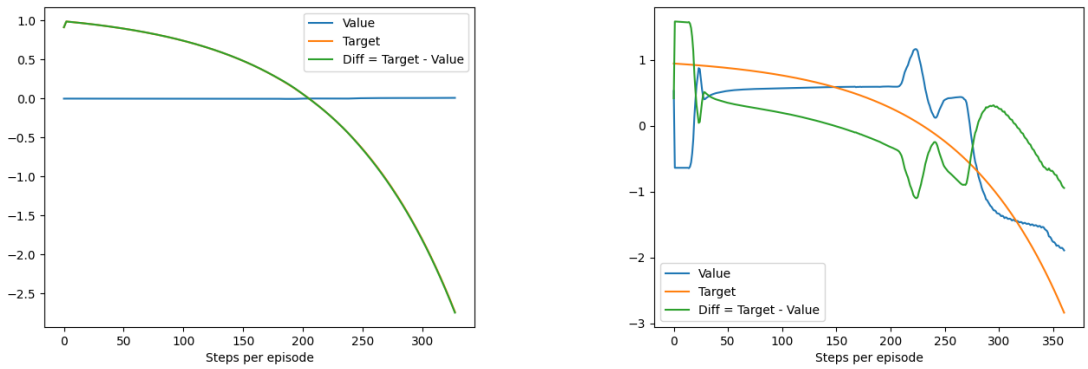


Figura 4.11: Rango de wd por episodio de entrenamiento - entre episodios 146 y 150

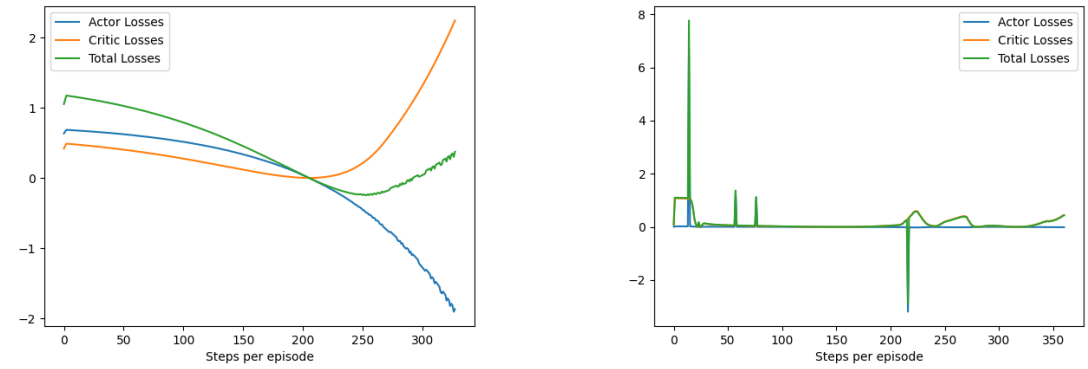
Como en ambos episodios el comportamiento es válido, se escoge el rango de pesos del episodio 146. Para comprobar el funcionamiento de la arquitectura neuronal, se muestran a continuación las gráficas correspondientes con la comparativa de las salidas reales y deseadas de la red así como las funciones de pérdidas, del episodio inicial y del episodio 146 (en el que se alcanza un comportamiento óptimo en comparación con el resto):



(a) Episodio 0 (b) Episodio 146

Figura 4.12: Valores de salida reales vs esperados - segunda versión optimización wd

En la Figura 4.12a se puede apreciar la gran diferencia (verde) entre la salida real y deseada de la red para el episodio inicial, que es disminuida considerablemente en el episodio 146 como se observa en la Figura 4.12b. Además, para este episodio 146, se ve en la Figura 4.13b cómo la función de pérdidas se sitúa en valores prácticamente nulos en la mayor parte de sus muestras, como no ocurría en el episodio inicial 4.13a.



(a) Episodio 0 (b) Episodio 146

Figura 4.13: Función de pérdidas - segunda versión optimización wd

Para validar los resultados obtenidos en el entrenamiento de esta segunda versión del controlador MPC en relación al error de distancia lateral, se procede a ejecutar el controlador con un valor de wd = 126 para después comparar el error de distancia lateral a lo largo de la trayectorias respecto a su versión inicial en la que wd = 1. En la Figura que se muestra a continuación se representan los resultados obtenidos:

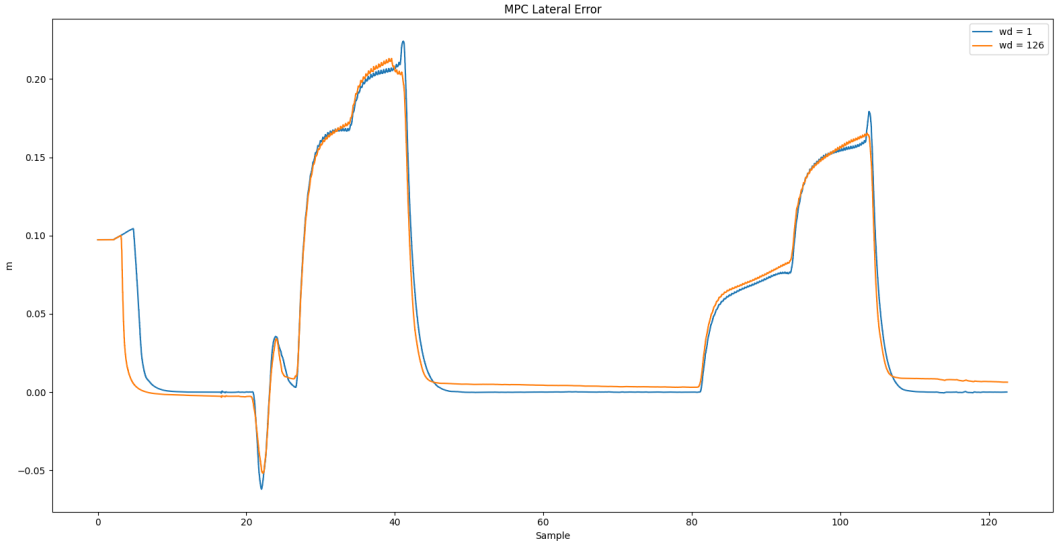


Figura 4.14: Comparación de error de distancia lateral tras optimización- segunda versión

En la Figura 4.14 se muestra la comparativa del error de distancia lateral en metros de la ejecución del controlador MPC con el valor de wd inicial escogido experimentalmente (wd = 1 en color azul) con respecto a uno de los valores del rango de wd óptimos estimados por la arquitectura neuronal Actor-Critic, en concreto wd = 126 (en color naranja).

Es admirable de observar cómo con este nuevo valor de wd = 126 obtenido por el sistema de aprendizaje neuronal, el comportamiento del controlador MPC sobre el vehículo consigue un mejor comportamiento del mismo en relación al error de distancia lateral sobre la trayectoria de entrenamiento.

En las muestras iniciales, el sistema nuevo (naranja) reduce ese pico inicial respecto al sistema inicial (azul), y además es capaz de llegar a error 0 más rápidamente respecto al original. Además, las oscilaciones entorno a la muestras número 20 y número 40, presentan comportamientos más suaves. También el error que se da entorno a la muestra 40, es reducido respecto a su versión inicial. Aunque entre las muestras 40 y 60 con esta nueva versión el error de distancia lateral sea ligeramente mayor al original, se ve una tendencia claramente descendiente hacia valores nulos, aunque más lenta que el original, que aún así, sería válido al situarse en valores de error de distancia lateral cercanos a 1 o 2 centímetros. Por último cabe destacar cómo en la muestra número 100 con este nuevo valor de wd se consigue reducir el pico de error del sistema original, llevando así a un comportamiento mucho más suave del sistema.

Por lo general queda validado este nuevo valor de wd = 126 como valor óptimo de peso de distancia lateral respecto a su valor original de wd = 1 por presentar un comportamiento mucho más suave en sus picos de error llevando así a una conducción mucho más confortable y segura para los pasajeros del vehículo.

4.2 Optimización del peso de error de yaw

Tras obtener unos resultados aceptables en la optimización del peso de error de distancia lateral se procede a optimizar el peso de error de yaw. Este proceso tuvo lugar únicamente con la segunda versión del controlador MPC en la que el peso asociado a dicho error (wy) tenía un valor inicial de wy = 1.

Como había sido comentado en la introducción, en la sección de Aprendizaje por Refuerzo 1.3, de la Figura 1.18, que corresponde con la primera versión del MPC, se extraía la conclusión de que la restricción de error de yaw para entrenar el sistema debía ser de 0,07 radianes (4º sexagesimales) para así reducir dicho error sobre la primera curva de la trayectoria de entrenamiento con la estimación de su valor de peso óptimo.

Atendiendo a la gráfica del error de yaw a lo largo de toda la trayectoria de la segunda versión del MPC de la Figura 4.15 (en la que wd = 1 e wy = 1), se extrae el dato de la nueva restricción de entrenamiento.

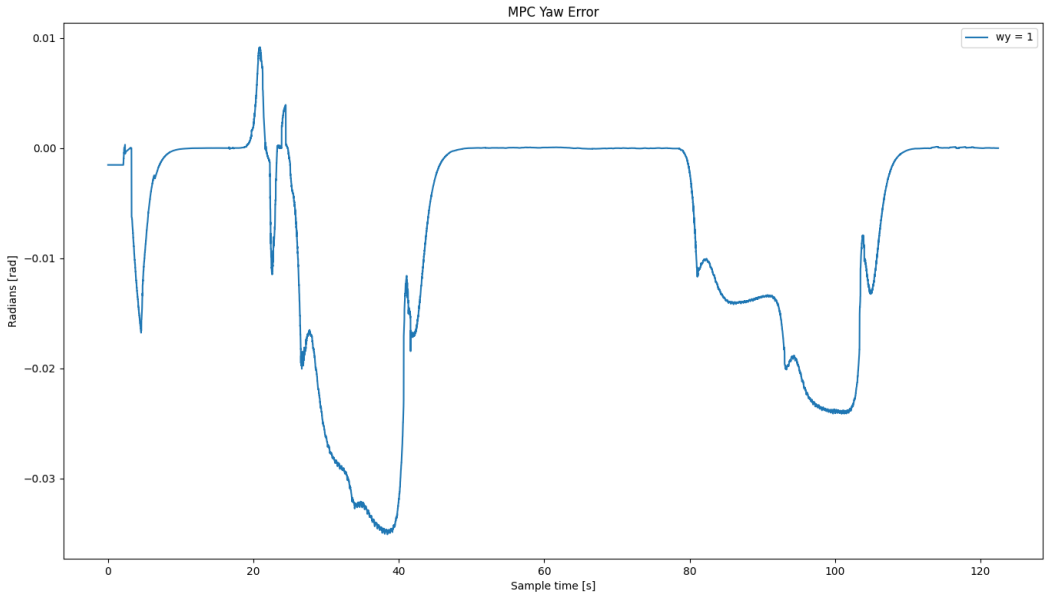


Figura 4.15: Error de yaw - segunda versión MPC

Como puede observarse en las muestras próximas a la número 40 de la Figura 4.15, se alcanza un error de yaw máximo de -0,034 radianes (-1,95º sexagesimales). El valor absoluto de este dato será empleado como nueva restricción de entrenamiento para esta segunda versión del MPC en relación a la búsqueda del rango de valores óptimos de wy que mejoren el comportamiento del vehículo en lo que se refiere al error de yaw sobre la trayectoria.

Para validar el funcionamiento del sistema de aprendizaje neuronal sobre este nuevo parámetro, se procede a entrenar el sistema con un valor inicial de peso de yaw negativo para demostrar cómo el vehículo inicialmente no consigue moverse por la trayectoria apenas unos metros hasta salirse de la misma y ver cómo progresa hacia valores óptimos de wy que consigan llevar al vehículo por la trayectoria de entrenamiento. Los parámetros que fueron empleados en este entrenamiento se muestran en la Tabla 4.3.

Parámetro	Valor
Learning Rate	0,001
wy inicial	-1
Variación de wy	±0,1
Tiempo de entrenamiento	30 minutos

Tabla 4.3: Parámetros de entrenamiento wy

Tras varios ensayos se comprobó que el sistema aprendía mucho más rápido y con mejores resultados con un valor de (*Learning Rate*) de $\delta = 0{,}001$. También mostró mejor comportamiento una variación de wy de $\pm$ 0,1 por iteración de la red Actor así como un wy inicial de -1. Con todos estos parámetros se consiguieron unos resultados aceptables en tan solo 30 minutos.

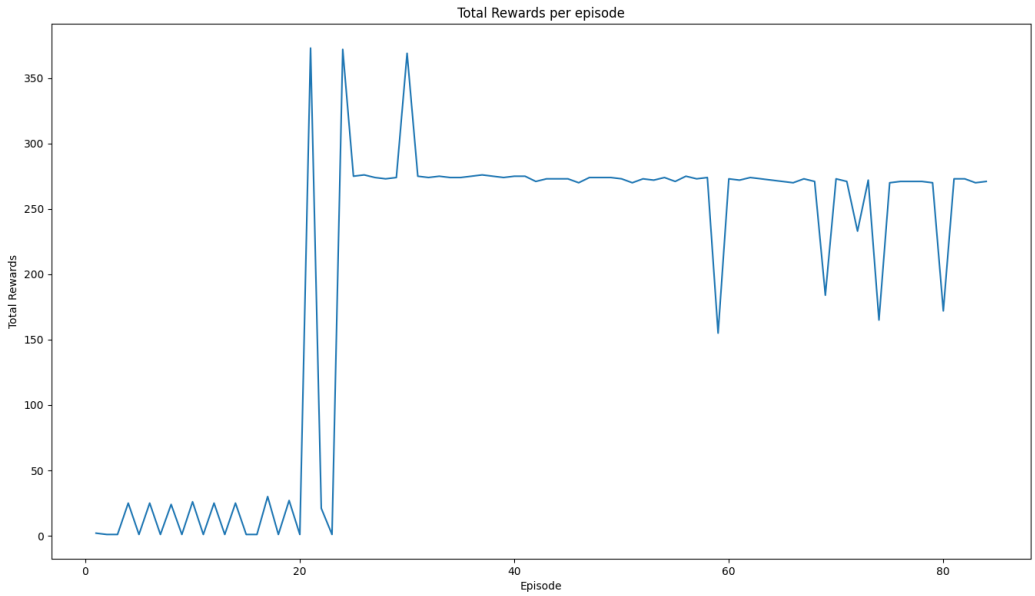


Figura 4.16: Rewards por episodio - optimización wy

En la Figura 4.16 se muestran los rewards por episodio de entrenamiento en la optimización del peso de error de yaw.

Hasta el episodio número 20 se observan muy pocos rewards por episodio ya que el vehículo está recibiendo valores negativos de wy que provocan que no pueda recorrer una distancia considerable sin salirse de la trayectoria. Conforme el sistema de aprendizaje va adquiriendo más experiencias pasadas consigue que los valores de wy evolucionen hacia valores positivos haciendo así que el vehículo sea capaz de moverse a lo largo de la trayectoria.

Los episodios en los que el vehículo sigue mayor parte de la trayectoria corresponden con el número 21, 24 y 30, atendiendo a los picos de rewards cercanos a 350 de la Figura 4.16. Además, se puede observar cómo a partir del episodio número 30 los rewards por episodio permanecen en valores prácticamente constantes entre 250 y 300 llegando incluso a valores inferiores en episodios cercanos al 60 y el 80. Esto demuestra que el mejor resultado no es alcanzado de nuevo en los últimos episodios del entrenamiento y que por lo tanto, atendiendo a los rewards máximos que tienen lugar en los episodios 21, 24 y 30 y el rango de valores de wy por episodio, según se muestra en la Figura 4.17, se deduce que a partir de cierto

rango de valores de wy los rewards obtenidos son menores y por lo tanto el sistema se comporta peor, pues el vehículo recorre menor parte de la trayectoria con valores más altos de wy.

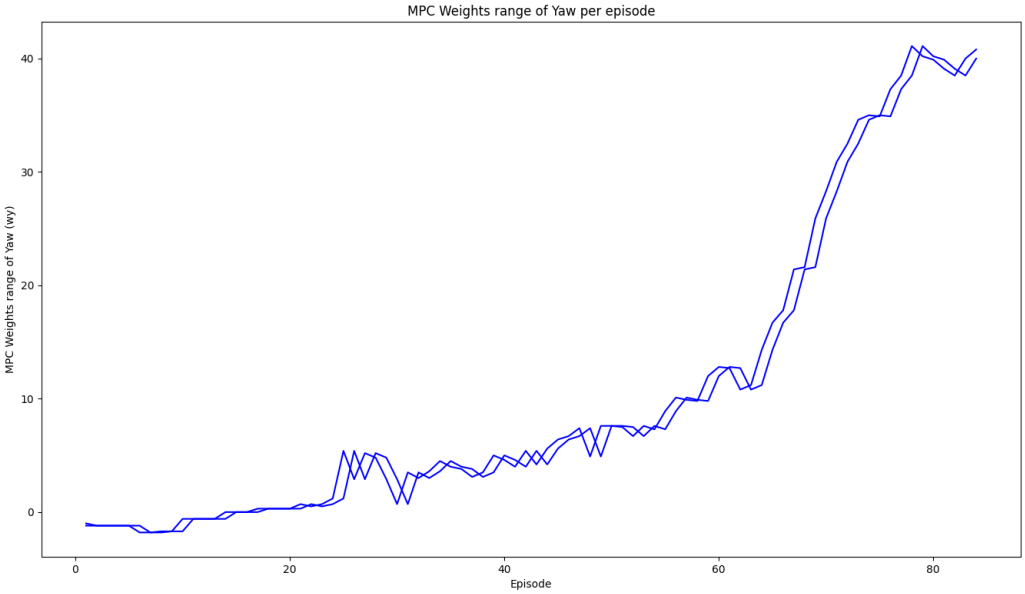


Figura 4.17: Rango de wy por episodio de entrenamiento

Para obtener el rango de valores de wy óptimos se procede como en el caso anterior de wd a observar la diferencia entre el valor de wy inicial y final de los episodios con mejor comportamiento y ver en cuál de ellos esta diferencia es menor ya que será el rango en el que el sistema de aprendizaje ha variado mucho menos los valores de wy, significando una cierta convergencia de datos. Para observar con más detalle esta diferencia de los episodios 21, 24 y 30, se procede a mostrar la gráfica ampliada entre los episodios 20 y 31 como se muestra en la Figura 4.18.

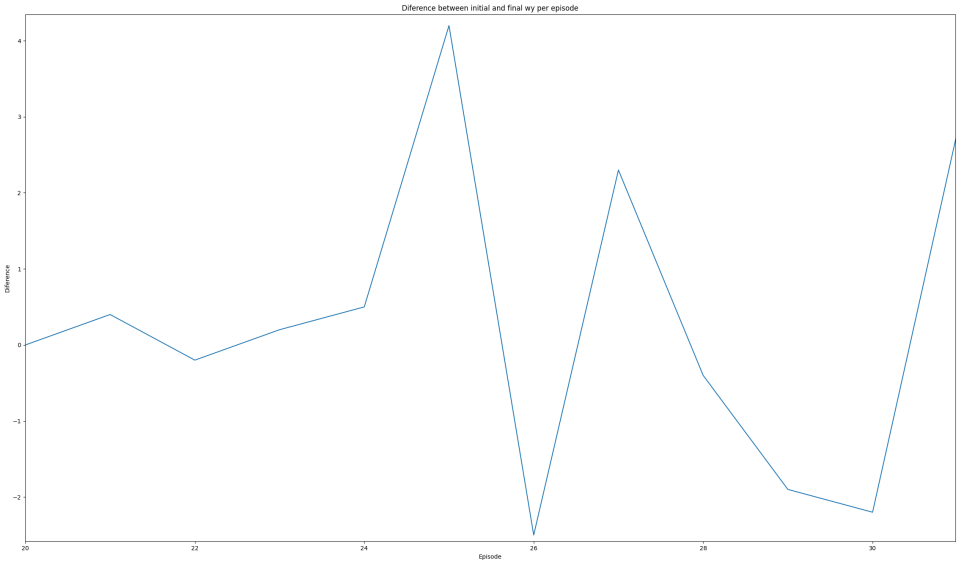


Figura 4.18: Diferencia de wy entre los episodios 20 y 31

De la Figura 4.18 se extraen las conclusiones de que en el episodio 30 ha habido una mayor divergencia de valores de wy en comparación con los episodios 21 y 24, ya que su diferencia entre wy inicial y final

en este episodio es de -2. Sin embargo, en los otros dos episodios, esta diferencia se sitúa cercana a 0,5 llegando así a unos datos que muestran mayor convergencia, ya que la variación entre el valor de wy inicial y final en estos episodios ha sido prácticamente nula.

Por lo tanto los episodios en los que ha resultado un mejor comportamiento del vehículo sobre la trayectoria en relación al error de yaw corresponden con el número 21 y 24. Para extraer el rango de valores de wy correspondientes a estos dos episodios se procede a representar la Figura 4.19 que es la misma que la Figura 4.17, pero ampliada entre los episodios 21 y 24 para visualizar mejor el rango de valores óptimos de wy.

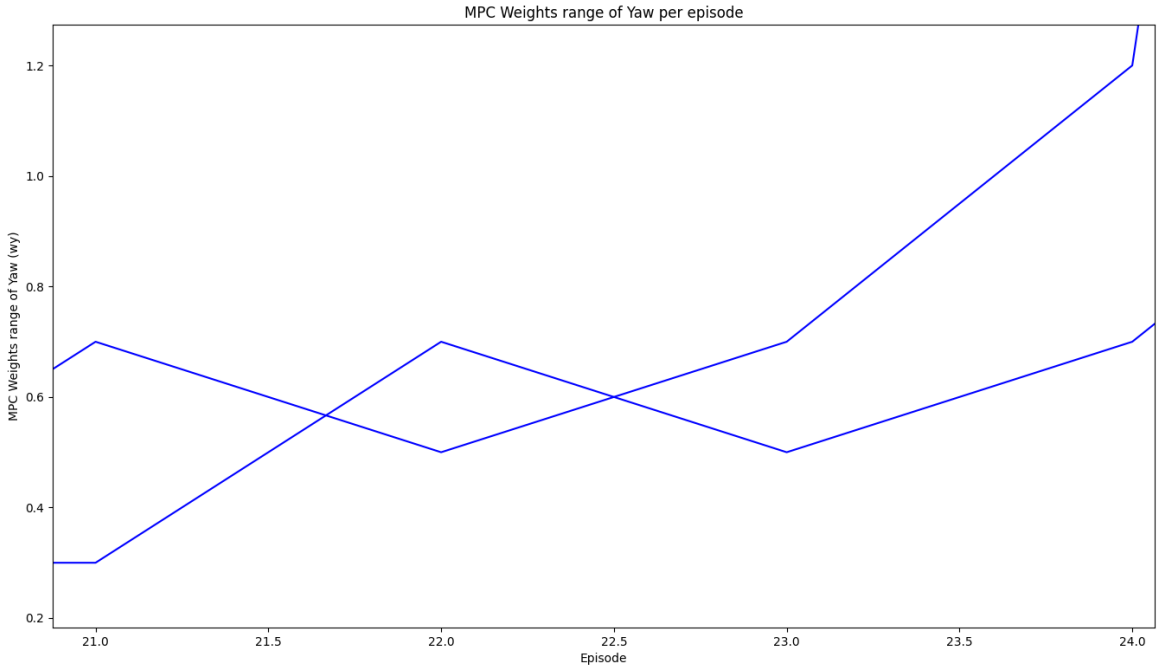


Figura 4.19: Ampliación del rango de wy entre episodios 21 y 24

De la Figura 4.19 se extraen como rangos de wy óptimos aquellos entre 0,3 y 0,7 para el episodio 21 y entre 0,6 y 1,2 para el episodio 24. Se comprobó que los valores que mostraban mejor comportamiento que el MPC original de esta versión (wy = 1) eran aquellos cercanos a este valor, por lo que se decidió por ejecutar el controlador MPC con valores de wy de 0,9, 1,1 y 1,2 ya que todos ellos pertenecían al rango del episodio 24, considerando por lo tanto dicho episodio como aquel en el que se dio el mejor comportamiento del vehículo en lo que respecta al error de Yaw.

Para escoger cuál de los valores del intervalo de wy del episodio número 24 será aquel que demuestre mejor comportamiento de error de yaw a lo largo de la trayectoria en comparación con la versión original del MPC (wd = 1 e wy = 1), se procede a mostrar en una misma gráfica el error de yaw en comparación con la versión original de wy = 1, cuyos resultados pueden apreciarse en la Figura 4.20 que se muestra a continuación:

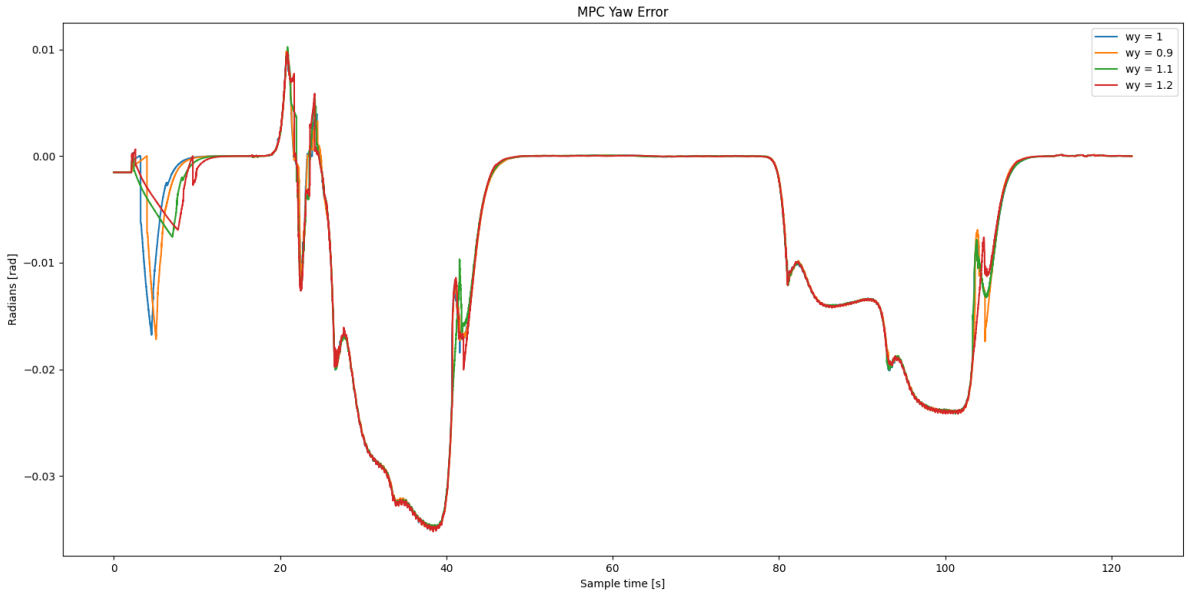


Figura 4.20: Comparativa de valores de wy óptimos

En la Figura 4.20 se observa claramente cómo el valor de wy = 1,2 es el que presenta mejor comportamiento en comparación con el resto. Para el caso de wy = 0,9 se observa un comportamiento similar que para el caso original del MPC en el que wy = 1, aunque se obtienen mejores resultados (menor error de yaw a lo largo de la trayectoria) para valores de wy ligeramente superiores a 1. Un valor de wy superior a 1,2 estaría fuera del rango de valores de wy óptimos mencionado anteriormente, por lo que se acepta el valor de wy = 1,2 como valor óptimo de peso de error de yaw para el controlador MPC. Para realizar un análisis más detallado de este peso (wy = 1,2) en comparación con el original (wy = 1), se procede a mostrar la gráfica de la Figura 4.21.

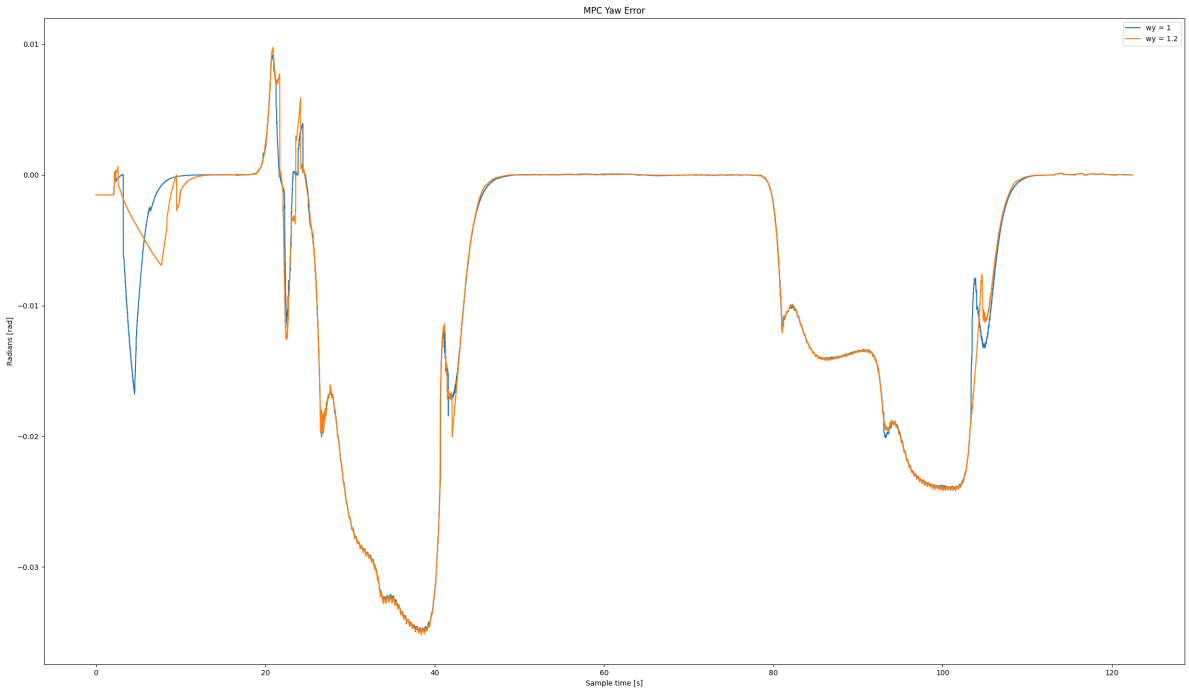


Figura 4.21: Comparación de error de yaw tras optimización de wy

En la Figura 4.21 se muestra en azul el error de yaw a lo largo de la trayectoria con el peso de error de yaw original (wy = 1) y en naranja con el valor de wy óptimo estimado por la arquitectura de aprendizaje neuronal (wy = 1,2).

Se puede observar claramente cómo este nuevo valor de wy consigue que el vehículo se mueva a lo largo de al trayectoria entre las muestras inicial y la 20 con un error de yaw mucho más pequeño y con un comportamiento más suave al reducir ese pico inicial en comparación con la versión original.

Aunque aparentemente entre las muestras número 20 y 40 el comportamiento del vehículo con este nuevo valor de wy no se vea mejorado en comparación con la versión original, se valida aun así este nuevo valor de wy por reducir considerablemente el error de yaw en las muestras iniciales y en aquellas cercanas a la 100, en la que el error de yaw adquiere un valor ligeramente inferior con una forma de onda prácticamente similar respecto a la versión original.

Tras los resultados obtenidos queda validado el valor de wy = 1,2 como peso de error de yaw óptimo para esta última versión del controlador MPC por presentar mejor comportamiento en líneas generales en referencia al error de yaw, donde se ven reducidos algunos valores máximos alcanzados en las muestras iniciales y finales y además no se muestra significativamente alterado en las muestras intermedias.

Capítulo 5

Conclusiones y trabajos futuros

Tras el análisis de los resultados obtenidos en este proyecto se puede concluir que el empleo de técnicas de aprendizaje por refuerzo para el ajuste de los parámetros de la función de costes de un controlador MPC queda validado y aceptado al mejorar significativamente su desempeño sobre el control de la trayectoria de un vehículo autónomo al reducir cuantitativa y cualitativamente tanto el error de distancia lateral como de yaw respecto a su versión original en la que dichos parámetros habían sido obtenidos de forma experimental.

En lo que respecta al error de distancia lateral, cabe mencionar la dificultad para llegar a una convergencia de la red neuronal y por lo tanto a una solución válida para el problema planteado por resolver. Fue bastante complicado el ajuste de los parámetros de la red neuronal para adecuar el modelo de aprendizaje planteado a la ejecución del controlador MPC para obtener unos resultados coherentes. En primer lugar, fueron planteadas varias restricciones de error sobre las que llevar a cabo el entrenamiento, deduciendo que los mejores resultados se obtenían con una restricción de error lateral de 20 centímetros que se daba en la primera curva pronunciada de la trayectoria de entrenamiento, para que el sistema fuera capaz de entrenar con datos de la primera recta y esta curva pronunciada, obteniendo así una solución generalista del problema. Tras numerosas simulaciones, finalmente se llegó a una de ellas en la que en 2 horas de entrenamiento el sistema logró llegar a unos resultados aceptables.

Cuando fue actualizada la versión del controlador MPC y hubo que entrenar de nuevo la red para obtener los nuevos pesos de distancia lateral óptimos, se adoptaron dos soluciones para optimizar el sistema de entrenamiento neuronal al observar que se perdía demasiado tiempo en ejecutar el controlador MPC junto con el algoritmo Actor - Critic y obtener unos resultados válidos en un periodo de tiempo aceptable. La primera de las soluciones, consistió en emplear la memoria gráfica del ordenador para realizar los cálculos complejos de entrenamiento de la red neuronal relacionados con las funciones de pérdidas y el descenso del gradiente, así como la actualización de los parámetros internos de la red (pesos y biases). La segunda solución, consistió en reducir el número de muestras por episodio de entrenamiento aumentando el tiempo de muestreo del sistema Actor - Critic a 100 ms, reduciendo así en 2 órdenes de magnitud de en torno a 30000 muestras por episodio que se obtenían inicialmente a 300 aproximadamente.

Con estas dos soluciones planteadas se consiguieron obtener resultados aceptables en relación al peso de error de distancia lateral de la nueva versión del MPC de forma mucho más rápida que en el caso anterior, obteniendo así un rango de valores de pesos asociados a este error que consiguieron reducirlo en la mayor parte de su trayectoria y provocar, desde un punto de vista de ingeniería de control, un desempeño del controlador MPC mucho más estable y preciso.

Por último con las mejoras incorporadas en el sistema de aprendizaje neuronal y con la última versión

del MPC, se procedió de nuevo a optimizar el valor del peso de error de yaw. Cabe destacar que el error máximo del que se partía en este caso era de apenas 0,034 radianes (1,95 grados sexagesimales), un error de yaw que es prácticamente imperceptible por el ser humano pero que aun así se propuso ser reducido lo máximo posible con la arquitectura neuronal desarrollada. En comparación con el peso de error de distancia lateral, en este caso, la red convergía bastante rápido y lograba alcanzar un rango de valores de pesos asociados a dicho error en poco más de media hora.

Como últimas conclusiones finales, mencionar que las redes de aprendizaje por refuerzo son bastante complicados de entrenar en comparación con sistemas de aprendizaje supervisado, ya que para cada experiencia nueva de entrenamiento se tienen que desarrollar demasiados episodios e ir obteniendo un comportamiento favorable del agente sobre el entorno para llegar de forma más o menos rápida a una convergencia de la red. Además, se requiere de bastante conocimiento previo para establecer una política óptima para el agente así como unos hiperparámetros aceptables para llegar a una rápida convergencia, siendo fundamental además el empleo de computación paralela con una tarjeta gráfica en el momento que el problema adquiere cierta complejidad.

El empleo de sistemas adaptativos en vehículos autónomos, como puede ser este, que consiste en ajustar los parámetros de la función de costes de un MPC, en un futuro vehículo autónomo serán fundamentales, debido a que el desgaste de los componentes mecánicos y electrónicos del vehículo provocará que su comportamiento en cuanto al seguimiento de trayectorias se vea afectado conforme vaya pasando el tiempo.

Trabajos futuros

Este proyecto parte del controlador de trayectorias MPC para un vehículo autónomo desarrollado previamente por otros investigadores del grupo INVETT [2]. Su objetivo fundamental fue mejorar lo máximo posible su comportamiento, quedando aun así muchas áreas de trabajo e investigación sobre las que proseguir en un futuro que son mencionadas a continuación:

- **Optimización del resto de pesos del MPC.** En este proyecto se estudiaron los pesos de error de distancia lateral y de yaw, quedando por ser optimizados los pesos de error de velocidad, error de steering y error de aceleración.
- **Implementación sobre vehículo autónomo real.** Todas las pruebas de este proyecto se realizaron sobre el entorno de simulación de vehículos autónomos CARLA. Una vez el MPC haya sido optimizado en su totalidad, podría ser probado sobre el vehículo automatizado Citroën C4 del grupo de investigación INVETT [2] y ver así como se comporta en cuanto a sus errores en una misma trayectoria su versión original respecto a la optimizada.
- **Nueva optimización sobre versión con velocidad variable.** El desarrollo de este proyecto se llevó a cabo a una velocidad constante del vehículo de 5 m/s para obtener así una primera aproximación de unos resultados aceptables. Para un futuro vehículo autónomo habría que obtener los valores de los pesos del MPC asociados a velocidades más altas de hasta 30 m/s para que el vehículo pudiese circular desde vías urbanas hasta vías interurbanas y autovías.
- **Optimización de pesos del MPC en tiempo real.** Como proyecto a largo plazo, se plantea el desarrollar un sistema que sea capaz de ir entrenando la red neuronal en tiempo real a la vez que se ejecuta el controlador MPC sobre el vehículo autónomo para así obtener un sistema completamente adaptativo.

Capítulo 6

Presupuesto

En este capítulo se procede a calcular el coste económico del proyecto.

6.1 Material hardware

El material hardware empleado en este proyecto se incluye en la Tabla 6.1 que se muestra a continuación:

COMPONENTE	PRECIO	AMORTIZACIÓN	USO	COSTE
Intel Core i7-8700 CPU	220 €	4 años	6 meses	27,50 €
Memoria 16 GB RAM	60 €	4 años	6 meses	7,50 €
Nvidia TITAN RTX	2.750 €	4 años	6 meses	343,75 €
Monitor LG 24"	120 €	4 años	6 meses	15 €
Teclado Logitech	12 €	4 años	6 meses	1,50 €
Ratón Logitech	6 €	4 años	6 meses	0,75 €
			TOTAL	**396 €**

Tabla 6.1: Costes hardware

6.2 Herramientas Software

El software empleado en este proyecto se incluye en la Tabla 6.4 que se muestra a continuación:

COMPONENTE	PRECIO	AMORTIZACIÓN	USO	COSTE
Software Ubuntu 18.04	0 €	N/A	N/A	0 €
Simulador CARLA	0 €	N/A	N/A	0 €
Editor Sublime Text 3	0 €	N/A	N/A	0 €
TeXstudio 3	0 €	N/A	N/A	0 €
Microsoft 365 Empresa	126 €	1 año	6 meses	63 €
			TOTAL	**63 €**

Tabla 6.2: Costes software

6.3 Mano de obra

CONCEPTO	PRECIO	HORAS	COSTE
Ingeniería	20 €/h	300 h	6000 €
Mecanografía	12 €/h	60 h	720 €
		TOTAL	**6720 €**

Tabla 6.3: Costes de mano de obra

6.4 Presupuesto total

CONCEPTO	COSTE
Material hardware	396 €
Herramientas software	63 €
Mano de obra	6720 €
TOTAL	**7179 €**

Tabla 6.4: Costes totales

El importe total del Trabajo de Fin de Grado *Controlador MPC adaptativo por DQN orientado a la conducción autónoma* asciende a un valor de 7179 euros sin incluir impuestos.

En Alcalá de Henares, septiembre de 2021.

Jesús Ángel Oliveros Fernández

Graduado en Ingeniería en electrónica y automática industrial

Bibliografía

[1] C. Zhao, L. Li, X. Pei, Z. Li, F.-Y. Wang, and X. Wu, "A comparative study of state-of-the-art driving strategies for autonomous vehicles," *Accident Analysis and Prevention*, vol. 150, p. 105937, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0001457520317577

[2] Intelligent vehicles and traffic technologies. [Online]. Available: http://www.invett.es

[3] B. Martin, "A basic working principle of model predictive control," Oct 02, 2009. [Online]. Available: https://es.m.wikipedia.org/wiki/Archivo:MPC_scheme_basic.svg

[4] H. Wang, B. Liu, X. Ping, and Q. An, "Path tracking control for autonomous vehicles based on an improved mpc," *IEEE access*, vol. 7, pp. 161 064–161 073, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8854189

[5] S. Perera. (2019, Aug 20,) An introduction to reinforcement learning. [Online]. Available: https://towardsdatascience.com/an-introduction-to-reinforcement-learning-1e7825c60bbe

[6] L. Weng. (2018, Feb 19,) A (long) peek into reinforcement learning. [Online]. Available: https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.html#key-concepts

[7] I. de Ingeniería del Conocimiento. Aprendizaje profundo por refuerzo. [Online]. Available: https://www.iic.uam.es/aprendizaje-profundo-por-refuerzo/

[8] M. Silva. (2019, Apr 28,) Aprendizaje por refuerzo: Introducción al mundo del rl. [Online]. Available: https://medium.com/aprendizaje-por-refuerzo-introducción-al-mundo-del/aprendizaje-por-refuerzo-introducción-al-mundo-del-rl-1fcfbaa1c87

[9] Aprendizaje por refuerzo: algoritmo q learning. [Online]. Available: http://www.cs.us.es/~fsancho/?e=109

[10] Y. Patel. (2017, Jul 31,) Reinforcement learning w/ keras + openai: Actor-critic models. [Online]. Available: https://towardsdatascience.com/reinforcement-learning-w-keras-openai-actor-critic-models-f084612cfd69

[11] L. Weng. (2018, Apr 08,) Policy gradient algorithms. [Online]. Available: https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html#what-is-policy-gradient

[12] Vista aérea vehículo. [Online]. Available: https://www.pngwing.com/es/free-png-bzyvp/download

[13] C. Yoon. (2019, Feb 6,) Understanding actor critic methods and a2c. [Online]. Available: https://towardsdatascience.com/understanding-actor-critic-methods-931b97b6df3f

[14] I. Grondman, M. Vaandrager, L. Busoniu, R. Babuska, and E. Schuitema, "Efficient model learning methods for actor-critic control," *IEEE transactions on systems, man and cybernetics. Part B, Cybernetics*, vol. 42, no. 3, pp. 591–602, Jun 2012. [Online]. Available: https://ieeexplore.ieee.org/document/6096441

[15] J. Peters and S. Schaal, "Natural actor-critic," *Neurocomputing (Amsterdam)*, vol. 71, no. 7, pp. 1180–1190, 2008. [Online]. Available: http://dx.doi.org/10.1016/j.neucom.2007.11.026

[16] A. Nandan. (2020, May 13,) Actor critic method. [Online]. Available: https://keras.io/examples/rl/actor_critic_cartpole/

[17] (2021, Feb 03,) Playing cartpole with the actor-critic method. [Online]. Available: https://www.tensorflow.org/tutorials/reinforcement_learning/actor_critic#1_collecting_training_data

[18] A. T. Kozak, "State-of-the-art in control engineering," *Journal of Electrical Systems and Information Technology*, vol. 1, no. 1, pp. 1–9, May 2014. [Online]. Available: https://dx.doi.org/10.1016/j.jesit.2014.03.002

[19] S. Kozak, "From pid to mpc: Control engineering methods development and applications," pp. 1–7, Feb 2016. [Online]. Available: https://ieeexplore.ieee.org/document/7438634

[20] M. L. Darby and M. Nikolaou, "Mpc: Current practice and challenges," *Control engineering practice*, vol. 20, no. 4, pp. 328–342, Apr 2012. [Online]. Available: https://dx.doi.org/10.1016/j.conengprac.2011.12.004

[21] M. Sanz Ausin. (2020, Apr 3,) Introducción al aprendizaje por refuerzo. parte 3: Q-learning con redes neuronales, algoritmo dqn. [Online]. Available: https://markelsanz14.medium.com/introducción-al-aprendizaje-por-refuerzo-parte-3-q-learning-con-redes-neuronales-algoritmo-dqn-bfe02b37017f

[22] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature (London)*, vol. 518, no. 7540, pp. 529–533, Feb 26, 2015. [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/25719670

[23] O. Nová?ek. (2020, Jun 15,) Connect x with dqn and pbt. [Online]. Available: https://medium.com/@novacek_48158/connect-x-with-dqn-and-pbt-be11915dd860

[24] A. Srinivasan. (2018, Jul 26,) Reinforcement learning: The business use case, part 2. [Online]. Available: https://medium.com/ibm-data-ai/reinforcement-learning-the-business-use-case-part-2-c175740999

[25] Lenguajes de programación para machine learning. [Online]. Available: https://aprendeia.com/lenguajes-de-programacion-para-machine-learning/

[26] Tensorflow. [Online]. Available: https://www.tensorflow.org/?hl=es-419

[27] Keras. [Online]. Available: https://keras.io

[28] Ubuntu. [Online]. Available: https://ubuntu.com

[29] Carla. [Online]. Available: https://carla.org

[30] T. Woods. (2017, Nov 28,) Gratis y con sello español: así es carla, el simulador de conducción autónoma más completo. [Online]. Available: https://www.technologyreview.es/s/9770/gratis-y-con-sello-espanol-asi-es-carla-el-simulador-de-conduccion-autonoma-mas-completo

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá